

UNIVERSITÉ DU QUÉBEC À MONTREAL

ANALYSE COMPARATIVE DU TEST EXPLORATOIRE ET DU
TEST SCÉNARISÉ : ÉTUDE EMPIRIQUE

MÉMOIRE
PRÉSENTÉ
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INFORMATIQUE

PAR
NAIMA ANKOUD

DÉCEMBRE 2007

UNIVERSITÉ DU QUÉBEC À MONTRÉAL
Service des bibliothèques

Avertissement

La diffusion de ce mémoire se fait dans le respect des droits de son auteur, qui a signé le formulaire *Autorisation de reproduire et de diffuser un travail de recherche de cycles supérieurs* (SDU-522 – Rév.01-2006). Cette autorisation stipule que «conformément à l'article 11 du Règlement no 8 des études de cycles supérieurs, [l'auteur] concède à l'Université du Québec à Montréal une licence non exclusive d'utilisation et de publication de la totalité ou d'une partie importante de [son] travail de recherche pour des fins pédagogiques et non commerciales. Plus précisément, [l'auteur] autorise l'Université du Québec à Montréal à reproduire, diffuser, prêter, distribuer ou vendre des copies de [son] travail de recherche à des fins non commerciales sur quelque support que ce soit, y compris l'Internet. Cette licence et cette autorisation n'entraînent pas une renonciation de [la] part [de l'auteur] à [ses] droits moraux ni à [ses] droits de propriété intellectuelle. Sauf entente contraire, [l'auteur] conserve la liberté de diffuser et de commercialiser ou non ce travail dont [il] possède un exemplaire.»

REMERCIEMENTS

Je tiens à remercier vivement Monsieur Robert Dupuis, professeur à l'université du Québec à Montréal et directeur de ce travail de recherche, d'avoir suggéré le sujet d'actualité de ce mémoire et accepté de le diriger. Je lui exprime ma profonde gratitude de m'avoir fourni assistance, conseils judicieux et encouragements.

Je voudrais également exprimer mes remerciements aux membres du jury, qui m'ont fait l'honneur d'avoir bien voulu évaluer mon travail.

J'aimerais aussi remercier sincèrement mes parents, mes frères et ma sœur pour leur soutien et leur confiance. Je tiens, en particulier, à remercier mon frère Hicham, qui m'a poussée à aller de l'avant et faire des études supérieures. Sans son soutien, ce travail n'aurait jamais pu être réalisé, je lui en serai éternellement reconnaissante.

Je tiens aussi à remercier mon amie Monia pour son appui, assistance et encouragement. Je me dois de dire un merci particulier à Mlle Laurence Loisel Dupuis qui a pris la peine de lire ce mémoire afin de m'aider à corriger les nombreuses fautes d'orthographe qui s'y étaient glissées.

Enfin, je remercie les autres personnes qui ont contribué de près ou de loin à ce travail, spécialement les étudiants de cours INF6160 de la session d'automne 2005, pour avoir accepté de participer à notre étude empirique.

Ce travail est dédié à ma mère.

TABLE DES MATIÈRES

LISTE DES TABLEAUX	viii
LISTE DES FIGURES.....	ix
LISTE DES ABRÉVIATIONS	x
RÉSUMÉ.....	xi
INTRODUCTION.....	1
CHAPITRE I.....	5
DÉFINITION DU SUJET DE RECHERCHE	5
1.1 ÉNONCÉ DE LA PROBLÉMATIQUE.....	5
1.2 MÉTHODE DE RECHERCHE.....	7
1.3 DÉFINITION DU PROJET.....	7
1.3.1 La motivation.....	8
1.3.2 La portée	8
1.3.3 L'objectif	8
1.3.4 Description des utilisateurs des résultats de la recherche.....	9
1.3.5 Les limites du projet	11
1.4 PLANIFICATION DU PROJET.....	11
CHAPITRE II.....	13
TEST SCÉNARISÉ : VUE D'ENSEMBLE.....	13
2.1 CONCEPTS DE BASE ET TERMINOLOGIE.....	13
2.1.1 Terminologie.....	13
2.1.2 Cas de test.....	14
2.2 TEST SCÉNARISÉ (SCRIPTED TESTING)	14

2.3	BRÈVE HISTOIRE DES TESTS.....	16
2.4	LES MODÈLES DE TEST	17
2.5	PROCESSUS DE TEST SCÉNARISÉ.....	20
2.5.1	La planification.....	23
2.5.2	Analyse et conception.....	24
2.5.3	Exécution et évaluation.....	29
2.6	CONCLUSION.....	29
CHAPITRE III	30
TEST EXPLORATOIRE : VUE D'ENSEMBLE	30
3.1	DÉFINITIONS	30
3.2	PROCESSUS DE TEST EXPLORATOIRE	34
3.3	TEST EXPLORATOIRE GÉRÉ PAR SESSION (SBTM).....	35
3.4	LES STYLES ET LES TECHNIQUES D'EXPLORATION.....	38
3.5	CONCLUSION.....	42
CHAPITRE IV	43
REVUE DE TRAVAUX RELIÉS.....	43
4.1	ÉTUDE DE ITKONEN ET RAUTIAINEN (2005).....	43
4.1.1	Les raisons d'utilisation du test exploratoire	44
4.1.2	Les modes d'utilisation du test exploratoire.....	45
4.1.3	Les bénéfices du test exploratoire.....	46
4.1.4	Les lacunes du test exploratoire.....	46
4.2	ÉTUDE DE PETTY (2005)	47
4.3	ÉTUDE DE LYND SAY ET VAN EEDEN (2003).....	49
4.4	ÉTUDE DE JAMES ET WOOD (2003)	53
4.5	ÉTUDE DE AMLAND ET VAGA (2002).....	56
4.6	SYNTHÈSE DES RÉSULTATS DES ÉTUDES PROPOSÉES	57
CHAPITRE V	60

L'ÉTUDE EMPIRIQUE.....	60
5.1 MOTIVATION DE L'ÉTUDE	60
5.2 LA STRATÉGIE DE L'EXPÉRIENCE	61
5.3 LE SCHÉMA DE L'EXPÉRIENCE.....	63
5.3.1 Objectifs et hypothèses de l'expérience	63
5.3.2 La conception expérimentale	64
5.3.3 Les instruments de l'expérience	64
5.3.4 Le traitement expérimental	65
5.4 ANALYSE DES RÉSULTATS DE L'EXPÉRIENCE	67
5.4.1 Analyse des résultats de test exploratoire	67
5.4.2 Analyse des résultats de TE et TS	70
5.5 CONCLUSION.....	73
CHAPITRE VI	74
CADRE CONCEPTUEL DE COMPARAISON.....	74
6.1 MISE EN CONTEXTE DE L'ÉTUDE COMPARATIVE.....	74
6.2 CADRE CONCEPTUEL DE COMPARAISON	76
6.2.1 Les caractéristiques d'utilisation	77
6.2.2 Les caractéristiques de gestion	78
6.2.3 Les caractéristiques techniques.....	78
6.2.4 Les caractéristiques du personnel	78
6.2.5 La productivité.....	79
6.3 CONCLUSION.....	81
CHAPITRE VII.....	82
ANALYSE COMPARATIVE DU TEST EXPLORATOIRE ET DU TEST	
SCÉNARISÉ	82
7.1 COMPARAISON SELON LES CARACTÉRISTIQUES D'UTILISATION	82
7.1.1 Les raisons d'utilisation.....	82
7.1.2 Les caractéristiques du logiciel.....	85

7.1.3	Le type d'environnement d'affaire	87
7.1.4	Les ressources financières	88
7.1.5	Le temps de test disponible.....	89
7.2	COMPARAISON SELON LES CARACTÉRISTIQUES DE GESTION.....	91
7.2.1	La planification.....	91
7.2.2	Le contrôle et le suivi de progression de test.....	93
7.2.3	La communication dans le projet de test	94
7.2.4	La relation avec le client.....	96
7.3	COMPARAISON SELON LES CARACTÉRISTIQUES TECHNIQUES	96
7.3.1	Les activités de test.....	97
7.3.2	Les risques du logiciel	100
7.3.3	La couverture de test.....	102
7.3.4	L'oracle de test	103
7.4	COMPARAISON SELON LES CARACTÉRISTIQUES DU PERSONNEL	106
7.4.1	Les caractéristiques du testeur.....	106
7.5	COMPARAISON SELON LA PRODUCTIVITÉ	108
7.7	CONCLUSION.....	113
CHAPITRE VIII	114
ANALYSE DES RÉSULTATS	114
8.1	ANALYSE DES RÉSULTATS THÉORIQUES	114
8.2	ANALYSE DES RÉSULTATS EMPIRIQUES	115
8.3	PISTES POTENTIELLES DE RECHERCHE.....	117
8.4	CONCLUSION.....	118
CONCLUSION	119
ANNEXE A	121
CADRE DE BASILI ADAPTÉ À LA RECHERCHE EXPLORATOIRE	121
ANNEXE B	124

PLAN DE TEST IEEE829.....	124
ANNEXE C	127
SOIRÉE DE TESTS.....	127
ANNEXE D	130
RAPPORT DE LA SESSION DE TEST.....	130
RÉFÉRENCES	131

LISTE DES TABLEAUX

Tableau 1.1 : Cadre de Basili adapté à la recherche exploratoire	7
Tableau 2.1 : La matrice de traçabilité.....	25
Tableau 3.1 : Grille des tâches de test exploratoire	34
Tableau 5.1 : ANOVA des données collectées de l'expérience.....	73
Tableau 7.1 : Le guide de sélection de l'approche de test	111

LISTE DES FIGURES

Figure 2.1 : Le pourcentage des erreurs d'analyse et de conception	17
Figure 2.2 : Modèle de test en V	18
Figure 2.3 : Les pratiques de test Agile	20
Figure 2.4 : Schématisation des documents de préparation de tests	22
Figure 2.5 : Spécification de Cas de Test	26
Figure 3.1 : Continuum repérant l'activité de test	33
Figure 3.2 : Cadre d'application de SBTM	37
Figure 3.3 : Heuristic Test Strategy Model	40
Figure 4.1 : Coût de documentation versus l'innovation dans le test	55
Figure 5.2 : Le traitement de test exploratoire.....	66
Figure 5.3 : Les résultats de test exploratoire	67
Figure 5.4 : Importance de défauts détectés avec le test exploratoire.....	68
Figure 5.5 : L'influence de l'expertise sur le test exploratoire	70
Figure 5.6 : Résultats des sujets aux TE et TS	71
Figure 5.7 : Représentation schématique de l'analyse ANOVA	72
Figure 6.1 : Le cadre conceptuel de comparaison.....	80
Figure 7.1 : Les éléments essentiels du test du logiciel	97
Figure 7.2 : Les activités de test scénarisé	98
Figure 7.3 : Les activités de test exploratoire	99

LISTE DES ABRÉVIATIONS

CDS	Context Driven School
IEEE	Institute of Electrical and Electronics Engineers
SBET	Session Based Exploratory Testing
SBTM	Session Based Test Management
SWEBOK	Software Engineering Body of Knowledge
UQÀM	Université du Québec À Montréal
TE	Test exploratoire
TS	Test scénarisé

RÉSUMÉ

Le test exploratoire (TE) est défini comme l'apprentissage, la conception et l'exécution simultanés des tests, tout à fait l'opposé du test scénarisé (TS) prédéfini. L'applicabilité de cette nouvelle approche ne cesse pas d'augmenter dans l'industrie du test de logiciel. Malgré cette expansion et le succès de quelques entreprises qui s'ouvrent dans le domaine de développement du logiciel dans ses expériences d'adoption et d'utilisation de TE, les contextes et les facteurs favorables pour l'adoption de l'approche dans une méthodologie de test ne sont pas toujours bien établis. L'absence des preuves claires de sa productivité annoncée par quelques praticiens dans la littérature s'ajoute à la problématique.

Ce travail est une étude exploratoire visant deux objectifs. Premièrement, étudier et analyser les contextes favorisant l'utilisation de TE comme une méthodologie primaire de test à la place des tests scénarisés en élaborant une analyse comparative entre le TE et le TS. Deuxièmement, évaluer sa productivité dans une étude empirique par rapport au TS. Nous avons élaboré un cadre conceptuel de comparaison dans lequel nous avons identifié cinq dimensions:

- Les caractéristiques d'utilisation : les raisons de l'utilisation, les caractéristiques du logiciel, le type d'environnement d'affaires, les ressources financières et le temps disponible pour les tests;
- Les caractéristiques de gestion : la planification, le contrôle et le suivi des tests, la communication dans le projet de test et la relation avec le client;
- Les caractéristiques techniques : les activités de test, l'oracle de test, les risques du logiciel et la couverture de test;
- Les caractéristiques du personnel : les caractéristiques des testeurs, la culture de l'organisation;
- La productivité : le nombre de défauts détectés, l'importance de défauts détectés.

Ce cadre a été utilisé comme base dans l'analyse comparative du TE et du TS. Dans cette analyse, nous avons comparé une approche disciplinée de TS guidé par les patrons de documentation IEEE 829 et une approche libre, semi planifiée de TE représentée par l'approche Session Based Exploratory Testing (SBET). Dans cette comparaison, la productivité a été évaluée par le biais d'une étude empirique que nous avons mise en œuvre, dans les laboratoires informatiques de L'UQAM. Malgré les limites du contexte de cette étude empirique, nous avons pu dégager quelques conclusions utiles.

Les résultats permettent de montrer que certains facteurs de contexte du projet de test peuvent empêcher l'utilisation de TE comme une méthode principale de test. Nous avons conclu que l'absence de contrôle de couverture de test restreint en plus le type des projets où le TE pourrait être utilisé. Aussi, l'expertise et les qualifications nécessaires pour exécuter le TE pourraient empêcher son utilisation dans les projets de tests où ces qualifications sont manquantes. Les résultats de l'étude empirique ont supporté l'hypothèse relative à l'importance des défauts détectés. D'autres recherches quantitatives sur la productivité de TE sont nécessaires, dont ce travail pourra servir comme point de départ.

Mots-clés

Test, Test scénarisé, Test exploratoire, Session Based Exploratory Test (SBET).

INTRODUCTION

La croissance rapide de l'utilisation des systèmes informatisés dans tous les domaines donne une importance cruciale au problème des anomalies informatiques. De nos jours, les ordinateurs aident les humains dans la plupart des aspects de notre vie quotidienne, et les applications informatiques sont rendues plus puissantes et complexes. À cet effet, l'importance de produire du logiciel de grande qualité et robuste est devenue primordiale (Osterweil, 1996). Or, la forte demande de produits logiciels de qualité fait que les organisations sont en quête continue de moyens pouvant leur permettre de produire de la qualité dans les temps et dans le cadre du budget. Un des moyens est le test du logiciel.

Dans les modèles traditionnels de développement, le test est un processus important. Il soutient l'assurance qualité en fournissant des informations sur la qualité du logiciel développé ou modifié. L'activité de test dans ces modèles est extrêmement laborieuse et demande des ressources importantes, allant de 50% à 60% du coût de développement (Perry, 2000). Cependant, la concurrence sans cesse croissante oblige les entreprises à s'adapter aux changements du marché dans des cycles de développement plus courts. Tester un logiciel dans telles conditions n'est plus une tâche facile, s'effectue souvent sous la pression de temps et de budget. Cela force l'industrie informatique à chercher de nouvelles méthodes efficaces de test pour épargner temps et argent et en produisant des logiciels de qualité.

Une nouvelle pratique de test, qui a fait son apparition à la fin des années 80, avait remis en cause la vision traditionnelle des tests. Cette pratique se définit comme l'apprentissage, la conception et l'exécution simultanée des tests. Cette dernière est tout à fait l'opposé de la méthode habituelle et scénarisée de test, nécessitant la spécification des cas de tests détaillés et des résultats prévus avant l'exécution de tests.

Au début, cette nouvelle pratique a été confondue avec le test ad hoc, qui est trop souvent le synonyme d'un processus improvisé, intuitif pour chercher les défauts du logiciel (Bach, 2003). En 1990 un groupe d'experts en tests, connu sous le nom anglophone Context Driven School (CDS) « Test piloté par le contexte » a adopté le terme anglophone Exploratory Testing « test exploratoire » pour décrire et nommer ce type de test, qui est décrit d'abord par Kaner (1988).

Depuis son apparition, le test exploratoire (TE) a été présenté comme une innovation dans l'industrie du test. Une innovation pouvant garantir un niveau d'efficacité de l'activité de test qui ne pourrait pas être réalisé par le test scénarisé seul (TS) (Bach, 2003; Kaner et Tinkham, 2003a). Cependant, quelques praticiens ne le voient pas de la même façon, et considèrent le TE comme un test ad hoc déguisé ou enveloppé sous le terme scientifique « exploration » (Black, 1999). Bach (2003) ne nie pas le fait que le TE est une pratique habituelle, utilisée souvent inconsciemment par n'importe quel testeur qui utilise sa créativité pendant l'activité de test. L'innovation de l'approche selon lui est d'améliorer la façon avec laquelle ce testeur effectue ses tests en se basant sur des techniques et des modèles scientifiques d'explorations. À cet effet, les praticiens et les chercheurs se sont penchés sur l'amélioration de ces techniques, dans le but de faire du TE une discipline apprise comme n'importe quelle activité intellectuelle. Enfin, avec la tendance actuelle des méthodes agiles, le TE a pu trouver sa place dans l'industrie du test (Bach, 2003). Les « agiles » ont adopté le TE compte tenu de sa grande capacité d'adaptation avec les pratiques agiles (Kohl, 2005).

Les grandes organisations de développement du logiciel ont commencé à considérer le TE comme une approche valide de test. Microsoft a utilisé un type structuré et documenté de TE implémenté par Bach (1999) pour tester et certifier une application pour la compatibilité et la stabilité avec Windows 2000. D'autres entreprises s'ajoutent continuellement, parce qu'elles cherchent des méthodes plus rentables de test. Toutefois, chacune d'elles adopte l'approche d'une façon personnalisée avec les contraintes de son contexte.

Malgré le succès obtenu par quelques entreprises dans l'implantation de TE, les contextes favorables pour utiliser et implémenter l'approche ne sont pas encore bien établis dans la

littérature. Nous nous sommes fixé comme objectif dans ce travail de recherche à explorer ces contextes en analysant les études de cas et les expériences professionnelles publiées récemment. Nous avons procédé à une analyse comparative entre les deux approches de test, le TS et TE, pour faire ressortir les facteurs favorables pour utiliser chacune d'elles.

Nous avons procédé aussi à une étude empirique de la productivité de TE annoncée dans la littérature, surtout par les deux concepteurs de l'approche (Bach, 2003 ; Bach et Kaner, 2004). Nous avons évalué la productivité de l'approche en termes du nombre et de l'importance des défauts qu'elle peut détecter. Malheureusement, les limites de contexte où s'est déroulé notre étude empirique ne nous a pas permis de tirer des conclusions fiables et généralisées sur cette productivité. Cependant, nous avons pu dégager quelques indications utiles.

Dans la réalisation de ce travail de recherche, nous avons dû confronter le problème de la pénurie des travaux de recherches et de la littérature qui traite le sujet de TE. Malgré que cette approche ait été introduite dans le domaine de test logiciel il y a plus de vingt ans, elle n'a pu que récemment attirer l'intérêt des chercheurs autres que les membres de CDS (Itkonen et Rautiainen, 2005). L'absence des connaissances à ce sujet nous a obligée à avoir recours à notre jugement dans cette analyse comparative, en s'appuyant sur notre compréhension de tous les éléments reliés à l'activité de test. Notre approche a été de s'appuyer sur la littérature, ainsi que sur divers concepts théoriques.

Dans le cadre de cette recherche, nous procédons par une étude exploratoire, puisque les connaissances dans le domaine que nous traitons dans ce travail de recherche ne sont pas encore bien établies.

Ce mémoire est composé de huit chapitres. Dans le chapitre I, nous définirons notre sujet de recherche, la problématique, la motivation, la portée, l'objectif, les utilisateurs et les limites du projet selon la méthode de (Abran et al., 1999), que nous avons adoptée pour la mise en œuvre de ce travail de recherche. Dans le chapitre II, nous présenterons une vue d'ensemble de TS afin d'identifier ses éléments fondamentaux et de comprendre ses points de différence

avec le TE. Dans le chapitre III, nous présenterons une vue d'ensemble de TE. Puis dans le chapitre IV, nous proposerons une revue de littérature de quelques études de cas et expériences d'utilisation de TE dans l'industrie de test. Le chapitre suivant décrit l'étude empirique que nous avons menée en laboratoire. Nous présenterons dans le chapitre VI, le cadre conceptuel de comparaison que nous avons développé dans le cadre de notre recherche. Le chapitre VII présentera notre analyse comparative de TE et TS. Nous présenterons des suggestions et les contextes favorables pour utiliser le TE comme une méthodologie primaire de test. Le dernier chapitre porte essentiellement sur les résultats d'analyse, l'interprétation de ces résultats, et les perspectives futures de recherche. Enfin nous terminerons notre rapport avec une conclusion.

CHAPITRE I

DÉFINITION DU SUJET DE RECHERCHE

1.1 Énoncé de la problématique

L'apparition de test exploratoire (TE) dans le domaine du test du logiciel, a suscité beaucoup de controverse quant à sa validité et à son efficacité. Pour certains praticiens, le test exploratoire est toujours le test ad hoc déguisé sous le terme scientifique « exploration » (Black, 1999). Selon Black (1999), le TE ne diffère pas de la technique « Error Guessing » proposée par Myers (1979) qui a toute fois précisé qu'il ne s'agit pas d'une technique de test, mais seulement de l'intuition. Selon Bach (2003), le TE est une approche valide de test, tout comme le test scénarisé (TS), et il pourrait être dans certaines situations plus productif que le TS. La différence sur la reconnaissance et la valeur de TE, qui est le chef d'œuvre et l'invention de la pensée Context Driven School (CDS), a lancé un débat entre les intervenants préconisant les principes de cette école de pensée et les adhérents de la discipline.

La philosophie de la pensée CDS est d'établir une relation consciente, explicable et autocritique entre le processus de test et le contexte de test. Autrement dit, le testeur devrait ajuster ses solutions convenablement au problème courant et être capable de contrôler son processus de test et d'apporter des changements au moment voulu pendant le processus. En 2001, les trois membres fondateurs de CDS, Kaner, Bach et Pettichord ont formulé les sept principes clés de leur discipline, et ils les ont publiés dans le premier livre portant sur les pratiques et les idées de ce groupe (Bach et al., 2002). Il s'agit des principes décrits ci-dessous (traduit de Bach et al, 2002, p., 261)

- La valeur de n'importe quelle pratique dépend de son contexte;
- Il y a de bonnes pratiques dans un contexte, mais il n'y a aucune meilleure pratique;

- Des personnes qui collaborent entre elles sont les parties les plus importantes de n'importe quel contexte de projet de test;
- Les projets se déroulent souvent d'une manière imprévisible;
- Le produit est une solution à un problème. Si le problème n'est pas résolu, le produit ne pourra pas fonctionner;
- Un bon test logiciel est un processus intellectuel comportant plusieurs défis;
- Seulement par les compétences qui s'exercent d'une façon coopérative dans tout le projet, nous sommes capables de prendre les bonnes décisions au bon moment, pour tester efficacement le produit.

Par contre, la discipline met l'accent sur des principes différents de ceux cités ci-dessus. Entre autres, la normalisation, la documentation et le formalisme du processus constituent les éléments clefs du processus de test (Thompson, 2003). Alors, il ressort des principes de deux écoles que le CDS accentue le rôle de la personne, ses qualifications et la collaboration entre les intervenants dans le projet de test, tandis que, la discipline accentue le rôle du processus, sa gestion et sa mesure dans le projet de test. En d'autres termes, le test devrait être prédictible, enchâssé dans un cadre de travail planifié et documenté dont la norme de documentation IEEE 829 pourrait constituer une partie intégrante (Swebok, 2004).

Les défenseurs des deux approches de test, le TS et le TE, ont lancé un débat qui oppose le formel contre l'informel, les procédures contre la liberté, l'uniformité contre la créativité, et le contrôle contre l'efficacité. Les auteurs et les praticiens de CDS annoncent que le TE est une approche productive qui pourrait constituer une méthode principale et indépendante de test (Bach, 2003; Kaner, 2006). Toutefois, ces mêmes praticiens n'ont pas identifié les contextes favorables d'utilisation de TE.

De plus, ils n'ont pas proposé des preuves concrètes de sa productivité largement annoncée dans la littérature, à part quelques anecdotes et expériences vécues élaborées dans des contextes personnels (Bach, 2003).

1.2 Méthode de recherche

Dans le cadre de cette recherche exploratoire, la démarche méthodologique suivie est basée sur le cadre de Basili (Abran et al. 1999), adapté aux études de cas de type exploratoire. Ce travail est inclus sous le thème de recherche exploratoire, compte tenu du fait que les problèmes soulevés sont peu abordés dans la littérature. Nous essayons de clarifier la base de connaissances sur le TE, et de trouver des pistes futures de recherche.

Le cadre proposé est composé de quatre étapes principales : la définition, la planification, l'opération et l'interprétation. Chaque phase décrit les activités et/ou les livrables à entreprendre afin d'atteindre les objectifs de cette recherche. Un modèle de ce cadre est présenté dans le tableau 1.1. La définition du projet est présentée dans la section qui suit. La structure suivie dans ce travail de recherche est proposée dans l'annexe A

Tableau 1.1 : Cadre de Basili adapté à la recherche exploratoire (Abran et al., 1999)

1. Définition			
Motivation	Objet	Objectif	Utilisateurs
2. Planification			
Étapes du projet	Intrants	Livrables du projet	
3. Opération			
Analyse des documents	Feedback des praticiens	Modèle proposé	
4. Interprétation			
Contexte	Extrapolation	Travaux futurs	

1.3 Définition du projet

La définition de la recherche à partir du cadre a permis d'établir la motivation, la portée, les objectifs de recherche, les utilisateurs des résultats de cette recherche et la planification du projet de recherche.

1.3.1 La motivation

La motivation principale de ce travail de recherche est d'explorer les apports du TE. Il s'agit aussi d'explorer l'ampleur de la divergence entre les deux écoles de pensées, la discipline et le CDS, par le biais d'une étude comparative approfondie des mécanismes et des techniques utilisés dans le TS et dans le TE. La productivité du TE est fortement annoncée par quelques praticiens, ce qui nous a aussi motivée à entreprendre une étude empirique pour évaluer la validité de ces annonces. Nous voulons par cette recherche contribuer à améliorer le processus de choix de l'approche convenable de test, en développant un guide d'évaluation des facteurs de contexte de projet de test. Ce guide va permettre de clarifier les contextes favorables pour utiliser le TE comme une méthodologie primaire de test.

1.3.2 La portée

Cette étude traite le test dynamique du logiciel selon un procédé boîte noire. Elle porte sur la comparaison de deux approches de test : le TE et TS.

1.3.3 L'objectif

L'objectif de notre étude est de présenter un ensemble de faits théoriques et expérimentaux qui peuvent justifier une réponse à notre question de recherche principale :

- *Est-ce que le test exploratoire pourrait remplacer le test scénarisé? Si oui, dans quels contextes?*

Pour répondre à la question principale de la recherche, nous essayons d'abord de trouver des réponses aux questions secondaires suivantes :

1. Quels sont les facteurs de contexte de test qui influencent le choix de l'approche de test?
2. Parmi ces facteurs, quels sont les facteurs de contexte de test favorables pour utiliser le TE?

Pour répondre à toutes ces questions, nous avons effectué une analyse comparative de TE par rapport au TS en utilisant un cadre conceptuel de comparaison développé dans le cadre de ce

travail de recherche. Ce cadre inclut des critères ou des facteurs de comparaison formulés à partir des résultats des études de cas de TE proposées dans la littérature, et en s'inspirant du cadre de comparaison proposé par Turner et Boehm (2004). Cette analyse comparative va permettre de souligner les contextes favorables à chacune des deux approches de test. Par la suite, nous avons conclu sur les contextes où le TE peut remplacer le TS.

Dans le cadre de cette étude comparative, nous avons procédé à une étude empirique de la productivité de TE. À cet effet, nous avons réalisé une expérience dans les laboratoires informatiques de l'UQÀM. L'objectif principal de cette expérience était de vérifier la productivité de TE en termes de nombre et d'importance de défauts qui pourraient être détectés en utilisant cette méthode de test. De plus, cette étude empirique a porté sur la comparaison de la productivité de TE par rapport au TS, autrement dit, l'analyse de l'effet de la technique de test (TE, TS) sur la productivité de l'activité de test, en utilisant un échantillon composé de 56 sujets (les étudiants de cours INF6160 session de l'automne 2005). Chaque élément de l'échantillon a appliqué les deux techniques de test sur un programme choisi. Dans notre plan expérimental, les mêmes sujets sont utilisés dans les deux traitements. Ce type d'étude est qualifié de plan expérimental à un seul facteur à mesures répétées «Single Factor Experiments Having Repeated Measures on The Same Elements». Nous avons exploité et analysé les résultats de deux façons différentes : la première, l'exploitation des résultats de TE collectés de notre expérience que nous comparons avec les résultats quantitatifs proposés dans la littérature. La deuxième, la comparaison des résultats collectés des deux approches. Nous avons procédé à l'analyse de variance ANOVA proposé par Winer (1971, p. 105).

En général, notre étude vise la clarification des connaissances concernant le TE, et s'intéresse tout particulièrement à l'évaluation de sa contribution dans le domaine du test logiciel. Nous voulons présenter les faits et les arguments existant dans la littérature, accompagnés de nos déductions personnelles.

1.3.4 Description des utilisateurs des résultats de la recherche

Ce travail de recherche revêt de l'intérêt pour plusieurs types d'intervenants : les chercheurs

en terme de contribution à l'ouverture de nouvelles pistes de recherche; pour les praticiens et les entreprises en terme de productivité et rentabilité de l'activité de test et gains financiers; et les enseignants en terme d'identification des nouvelles matières pour le cours de test du logiciel.

1.3.4.1 Les chercheurs

Les résultats de ce projet de recherche pourront être utilisés par les chercheurs intéressés par l'étude de TE. Comme domaine de recherche, les contextes favorables pour utiliser le TE n'ont pas été étudiés. Ainsi, la productivité du TE en termes du nombre et de l'importance des défauts qui pourront être détectés a été peu étudiée empiriquement. D'où l'importance des pistes et des solutions proposées dans ce travail de recherche. Les résultats théoriques et empiriques de cette étude pourront être réutilisés dans d'autres répétitions empiriques et travaux futurs, afin d'enrichir les connaissances existantes sur le TE.

1.3.4.2 Les entreprises

Au niveau pratique, cette recherche s'inscrit dans un courant de préoccupation des praticiens et des entreprises dans le domaine de test du logiciel, qui voudraient adopter et adapter le TE dans leurs méthodologies de test. En effet, les praticiens et les entreprises veilleront avec plus d'importance sur la rentabilité de l'activité de test, en réduisant la durée du cycle de test et en diminuant les efforts consacrés aux tests. Le guide de sélection de l'approche de test résultant de l'analyse comparative entre le TE et le TS vise l'identification des contextes favorables pour utiliser chacune des deux approches de test, afin d'atteindre la rentabilité et la productivité voulues et de répondre aux besoins des praticiens et les entreprises.

1.3.4.3 Les enseignants

Ce travail est destiné aux enseignants par l'identification des nouvelles matières pour le cours de test du logiciel. Habituellement, la matière de ce cours traite seulement le TS, du fait qu'elle est la méthode habituelle de test dans l'industrie. Or, la croissance continue de l'adoption de TE justifie l'inclusion de celui-ci dans les cours de tests pour préparer les étudiants à répondre aux besoins de l'industrie. L'utilisation du guide de sélection élaboré

dans le cadre de ce travail serait bénéfique pour aider les étudiants à identifier et à différencier les contextes qui favorisent l'utilisation de chacune des deux approches de test.

1.3.5 Les limites du projet

Une limite importante du projet vient de la limitation de contexte de notre étude empirique. L'absence d'un contexte industriel où nous pouvons effectuer notre étude empirique, nous a poussée de faire cette étude dans un contexte académique, en utilisant des étudiants comme des sujets de l'expérience et un petit programme comme instrument de l'expérience au lieu d'utiliser un grand logiciel. Ceci, a influencé les résultats de l'expérience.

1.4 Planification du projet

Dans le but d'atteindre notre objectif principal, nous avons identifié les phases suivantes :

1. Nous proposons un modèle de processus de TS, en mettant l'accent sur les livrables de chaque étape et la documentation élaborée. Notre modèle suit le standard de documentation IEEE829 (1998), pour pouvoir contraster les deux pensées discutées dans ce travail à savoir: la discipline et le CDS, chacune est représentée par sa pratique, successivement le TS et le TE.
2. Nous proposons l'approche Session Based Test Management (SBTM) qui va représenter le TE dans l'analyse comparative de TE et le TS.
3. Nous étudions les études de cas et les expériences d'utilisations de TE dans l'industrie de test. Nous déduirons les facteurs qui ont influencé l'utilisation et l'adoption de TE dans la pratique de test.
4. Nous réalisons notre étude empirique dans les laboratoires informatiques de l'UQÀM et nous traitons les données collectées pendant cette expérience.
5. Développement de notre cadre conceptuel de comparaison qui va guider l'analyse comparative de TE et le TS.

6. Nous procédons à une analyse comparative des deux approches de test: le TS et le TE selon le cadre de comparaison élaboré et les résultats de l'étude empirique.
7. Nous discutons les résultats de l'analyse comparative. Puis, nous concluons les contextes favorables pour utiliser le TE comme une méthodologie primaire de test. Nous terminons par l'élaboration d'un guide de sélection de l'approche de test

Le chapitre I présente l'étape 2 « Planification » de cadre de Basili. L'étape 3 «Opération» de ce cadre, sera abordée dans les chapitres II, III IV, V, VI et VII. L'étape 4 «Interprétation» sera abordée dans le chapitre VIII.

CHAPITRE II

TEST SCÉNARISÉ : VUE D'ENSEMBLE

Ce chapitre présente les éléments nécessaires à la compréhension du test scénarisé (TS). Dans la première partie nous définissons le TS et nous présenterons un bref historique du test du logiciel. Nous faisons ensuite un survol rapide de quelques modèles de test. Par la suite, nous proposerons un modèle du processus de TS balisé dans les patrons de la norme IEEE 829 et nous terminons par une conclusion.

2.1 Concepts de base et terminologie

2.1.1 Terminologie

Dans cette section nous proposons quelques définitions et certains termes et concepts fondamentaux du test logiciel. Premièrement, nous définissons les termes: erreur, défaut et défaillance. Ces termes s'utilisent parfois d'une façon interchangeable pour décrire un mauvais fonctionnement dans le logiciel. Swebok (2004) a identifié une faute ou défaut (*Defect, Fault*) comme la cause d'un mauvais fonctionnement dans le logiciel et l'effet non-désiré observé comme défaillance (*Failure*). Autrement dit, les tests révèlent les défaillances causées par un défaut qui peut et/ou doit être enlevé. En fait, Swebok a adopté les définitions proposées par IEEE à ces termes:

- **Erreur (Error):** Action humaine produisant un résultat incorrect (IEEE 729).
- **Défaut (Defect, Bug, Fault):** une imperfection dans un composant ou un système qui peut conduire à ce qu'un composant ou un système n'exécute pas les fonctions requises (IEEE 729).
- **Défaillance (Failure):** Fin de la capacité du système ou du composant à effectuer la fonction requise ou à l'effectuer dans les limites spécifiées (IEEE 729).

2.1.2 Cas de test

C'est un ensemble de valeurs d'entrée, de pré-conditions d'exécution, de résultats attendus et de post-conditions d'exécution, développés pour un objectif particulier, tel qu'exécuter un chemin particulier d'un programme ou vérifier le respect d'une exigence spécifique (IEEE 610). La norme IEEE829 (1998) a défini un cas de test comme un document indiquant les entrées, les résultats prévus et les conditions d'exécution pour un article de test.

Selon Kaner (2003), un cas de test est une question élaborée pour obtenir une information sur le programme sous test. Cette information se révèle par l'exécution du test relié à cette question. Cet auteur a cité deux conséquences indirectes de sa définition: la première est que le cas de test devrait être capable de fournir une information utile au moment de l'exécution. De ce fait, selon l'auteur, un cas de test conçu au début de l'activité de test ne pourra pas révéler une information ultérieurement dans le projet de test, quand le logiciel deviendra plus stable. C'est l'un des désavantages du TS cité par Copeland (2004). La deuxième implication que le cas de test ne devrait pas nécessairement détecter un défaut, mais il suffit qu'il fournisse une information, qui souvent mène à la détection d'un défaut selon l'auteur.

L'élaboration de cette définition n'est pas arbitraire, mais proposée pour inclure le cas spécial de cas de test exploratoire (TE), et ce pour deux raisons : la première est que les cas de tests exploratoires se fondent sur l'élaboration des questions à propos du logiciel sous test (Kaner et Tinkham, 2003a). La deuxième est qu'un cas de test exploratoire ne devrait pas obligatoirement détecter un défaut mais il suffit qu'il révèle une information. Cette information pourrait être utilisée pour concevoir un autre cas de test, qui pourrait mener à la détection d'un défaut (Kaner et Tinkham, 2003a).

2.2 Test scénarisé (Scripted Testing)

C'est un test manuel qui se fait typiquement par un testeur junior, en suivant les étapes et les procédures conçus par un testeur senior (Bach et al., 2002). Ces mêmes auteurs ont souligné plus tard dans plusieurs reprises que le test scénarisé pourrait être automatisé (Kaner, 2006).

Selon Copland (2004) le test scénarisé est n'importe quelle activité de test, manuelle ou automatisée, basée sur la conception et l'enregistrement des scripts détaillés de tests avant l'exécution.

Dans un TS, la planification et la conception des tests se font avant leur exécution. Les cas et les scénarios de test, typiquement, mais pas nécessairement, sont définis tôt dans le cycle du développement du logiciel, selon le modèle du cycle de vie utilisé. On les prépare en se basant habituellement sur le document des spécifications des exigences du logiciel dans un procédé de test boîte noire, sans accès au code, ou sur la logique interne du programme, dans un procédé boîte blanche, avec accès au code, ou une combinaison des deux. Les cas de tests sont alors exécutés par un autre testeur que celui qui les a conçus, quand le logiciel devient disponible pour les tests.

Historiquement, le test scénarisé a émergé en tant qu'une phase dans le premier modèle de développement en cascade (Royce, 1970). Dans ce modèle, le développement est décomposé en plusieurs étapes séquentielles, dont chacune possède des critères d'entrée et de sortie précis, et des livrables bien déterminés. Alors le test est l'une de ces étapes, son rôle est d'évaluer si les exigences sont bien comprises et spécifiées (Validation) et que la conception est correctement implantée par le code (Vérification). Récemment, le TS a franchi des nouvelles dimensions que celles connues dans les années 70. Alors, n'importe quelle méthodologie de développement même les plus agiles peuvent l'utiliser, n'importe où la répétitivité, l'objectivité et l'auditabilité sont nécessaires et importantes dans le processus de test du logiciel (Copeland, 2004).

Selon Copeland (2004) la répétitivité signifie que les tests ou les procédures de tests sont suffisamment détaillées pour qu'ils puissent être exécutés par quelqu'un autre que leur auteur original. En ce qui concerne l'objectivité, elle signifie que la création de tests ne dépend pas seulement de la créativité et la compétence de son concepteur, mais qu'ils sont basés sur des principes de conception bien déterminés, dérivés à partir des exigences du logiciel, les cas d'utilisations et les standards à respecter dans le projet de test. Quant à l'auditabilité elle signifie la traçabilité bidirectionnelle entre les spécifications d'exigences et les cas de tests.

Cette traçabilité permet de mesurer formellement la couverture de test qui sera abordée dans les sections qui suivent.

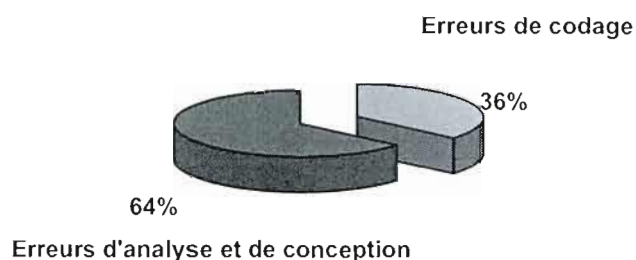
2.3 Brève histoire des tests

Myers (1979) a identifié le test logiciel en tant qu'une action d'exécuter un programme ou un système dans le but de détecter ses défauts. À l'époque, cette définition a été probablement la meilleure disponible. Elle reflétait les pratiques de cette période où le test apparaît seulement à la fin du cycle de développement visant principalement la détection des erreurs. Puis en 1988, la définition de test logiciel a changé en faveur de l'évaluation de la qualité du logiciel plutôt qu'un simple processus pour révéler les erreurs. Hetzel (1988) a défini le test comme une activité dont son but est d'évaluer et valider les fonctionnalités du logiciel. Récemment, le test est perçu comme une activité exécutée pour évaluer et améliorer la qualité du logiciel en identifiant ses défauts et ses problèmes (Swebok, 2004). Par cette définition Swebok ajoute l'amélioration de la qualité du logiciel à l'évaluation de la qualité et à la recherche des défauts. Cette méthode connue actuellement sous le terme de « *test préventif* » (Craig et Jaskiel, 2002; Swebok, 2004; Perry, 2000). Selon Swebok (2004) le test devrait encadrer l'ensemble du développement et de la maintenance et être en soi une partie importante de la construction du produit logiciel.

La philosophie de test préventif dicte que le test pourrait améliorer la qualité du logiciel, s'il apparaît assez tôt dans le cycle de développement. Il s'agit de la création de cas de tests pour valider les exigences et la conception, avant même le codage du logiciel. Cela permet de détecter les faiblesses potentielles et les erreurs dans les premières phases de développement et de réduire le coût de correction des défauts, sachant que plus l'erreur est découverte tôt dans le processus, moins elle coûte cher à corriger, et plus l'erreur se situe au début du cycle de développement, plus elle coûte cher à corriger ultérieurement dans le cycle (Boehm, 1981; Pressman, 1997). Selon Perry (2000) la plupart des erreurs détectées pendant la phase de test pourraient être attribuées à des erreurs d'analyse et de conception. Elles représentent approximativement, tel qu'illustré sur la figure 2.1, les deux tiers des erreurs faites pendant le développement du logiciel. En conséquence, la préparation du plan et des procédures de test scénarisé tôt dans le cycle de développement permettent de fournir des informations utiles

aux concepteurs, en identifiant les erreurs tôt dans le projet, comme les oublis ou les incohérences de conception, avant que ces erreurs se transforment à des défauts dans le code.

Figure 2.1 : Le pourcentage des erreurs d'analyse et de conception (Adapté et traduit de Perry, 2000, p.45)



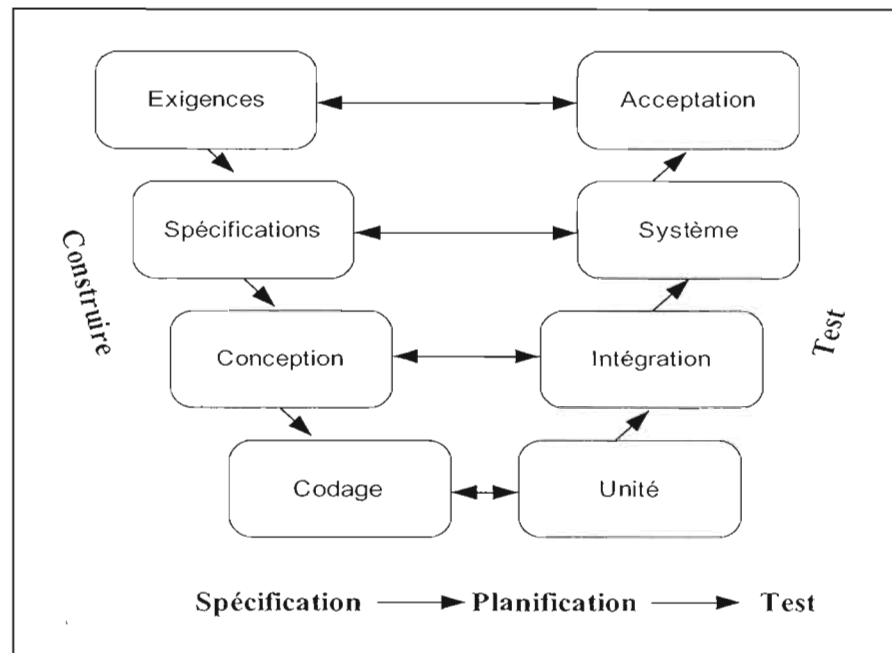
Dans les méthodes agiles, l'activité de test est incorporée dans chaque itération du processus de développement et constitue une partie intégrante essentielle de la construction du logiciel (Boehm et Turner, 2004). De ce fait nous croyons que les méthodes agiles satisfont aussi les aspects principaux de la définition proposée par Swebok (2004).

A part, une vue exploratoire, non traditionnelle propose le test comme une activité d'investigation et d'expérimentation. Selon Bach et Kaner (2004) le test est une investigation dont l'objectif est de révéler des informations sur la qualité du produit à tester. Cette définition vise principalement l'inclusion de TE qui se base sur l'investigation et le questionnement.

2.4 Les modèles de test

Le modèle en V constitue l'état de l'art, dans le domaine de test du logiciel. C'est le modèle le plus utilisé et traité dans la littérature de test (Craig et Jaskiel, 2002; Perry, 2000). Ce modèle tel qu'illustré à la figure 2.2, divise le processus de test en plusieurs niveaux, effectués conjointement avec l'implantation du logiciel. Il commence par le test de petits morceaux et avance vers des plus grands, reflétant différents points de vues de test à différents niveaux de détail.

Figure 2.2 : Modèle de test en V (Adapté et traduit de Pyhajarvi et al., 2003)



Ce modèle identifie des activités de test à chaque étape du cycle de vie. Le test unitaire est au niveau le plus bas dans la hiérarchie. L'objectif général de ce niveau de test est de trouver les défauts dans la logique, dans les données et dans les algorithmes de chaque module pris individuellement. Après le test unitaire, viennent les tests d'intégration, ces derniers sont effectués pour trouver les défauts d'interfaces entre les unités. En remontant dans la hiérarchie le niveau suivant les tests d'intégration est le test du système. Ce dernier est le processus de tester le système entier, afin de vérifier le produit par rapport aux spécifications des exigences. Finalement, le test d'acceptation prend place, afin de déterminer si le logiciel répond aux exigences et aux attentes du client.

La force de ce modèle est qu'il permet de planifier et de préparer les cas de test très tôt dans le cycle du développement, dès que l'abstraction des exigences est connue. Ce fait aide dans la détection des erreurs de conception et de spécification. Autrement, il permettrait de détecter les erreurs avant qu'ils se transforment à des défauts dans le code, c'est ce qu'est connu actuellement sous le terme de test préventif. Cependant, cette force ramène avec elle une

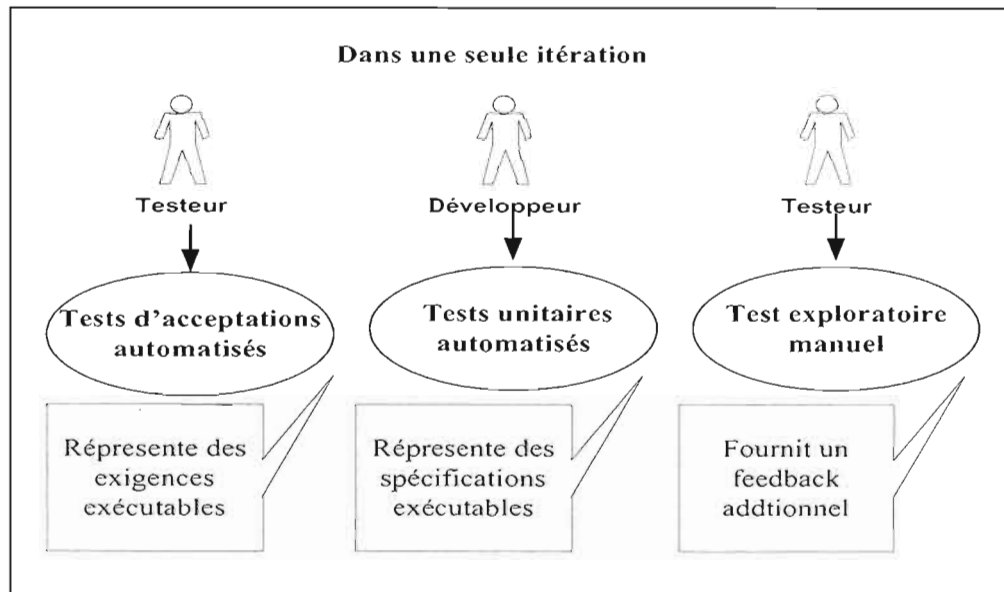
faiblesse importante. Cette faiblesse se traduit par la rigidité de modèle aux changements. En effet, souvent la documentation utilisée pour planifier et préparer les cas de tests n'est pas assez mûrie au début du projet de développement. Par conséquent, la possibilité que le logiciel se change ultérieurement dans le cycle de développement est forte probable. Ceci nécessite la mise à jour de tous les artefacts du test déjà conçus, qui est souvent très coûteux. Selon Bach et al (2002) les revues et les inspections pourraient être plus efficace dans la détection des erreurs des exigences et la conception que de concevoir des cas tests qui ne seront jamais exécutés.

Dernièrement avec l'apparition des méthodes agiles, le modèle de test en V ne peut plus suivre cette nouvelle tendance du développement (Pyhajarvi et al., 2003). En réaction, *le test Agile* a fait son apparition. C'est une méthode de test suivie dans les projets agiles de développement (Marick, 2003). Cet auteur a précisé que la communication et la collaboration entre les testeurs, les développeurs et le client constituent les principes essentiels du test agile. Le rôle du testeur n'est plus pareil à celui des modèles traditionnels, où le testeur s'occupe seulement de la vérification et la validation du logiciel développé. Cela nous amène vers un rôle plus constructif du logiciel, grâce aux informations et aux feedbacks utiles que le testeur pourrait fournir aux développeurs au fur et à mesure sur la qualité de l'application développée à chaque itération. Souvent le testeur utilise le TE pour tester le logiciel et fournir ce feedback (Pettichord, 2004)

La communication entre le testeur agile et le client est aussi très importante (Marick, 2003). Souvent le testeur devrait collaborer avec le client pour identifier les scénarios d'utilisation nécessaires à l'élaboration des tests d'acceptation. Cela fournira une revue implicite aux exigences et aide à bien visualiser le logiciel. En fait, le testeur peut assurer un feedback très tôt sur quelques aspects du logiciel tels que l'utilisabilité et d'autres fonctionnalités aux développeurs.

Selon Hendrickson (2005) le testeur a deux rôles dans le test Agile: testeur et designer. Les pratiques de test agile peuvent être résumées sur la figure ci-dessous :

Figure 2.3 : Les pratiques de test Agile (Adapté et traduit de Hendrickson, 2005)



Tel qu'illustré à la figure 2.3, le TE constitue une partie intégrante et essentielle de test agile.

2.5 Processus de test scénarisé

Pour faire la différence entre le TS et le TE, nous présentons dans cette section un processus de TS, sans spécifier une méthodologie précise. Nous nous intéressons seulement aux aspects qui nous permettront de mener notre étude comparative. Nous avons choisi de suivre le standard IEEE 829. Selon Copland (2004) cette norme de documentation est la meilleure dans le domaine du test qui peut illustrer les aspects principaux de l'approche scénarisée. Ce choix a été influencé aussi par nos besoins de contraster une vue scénarisée, disciplinée et formelle avec une autre exploratoire et libre.

La norme de documentation IEEE 829 de test du logiciel est une tentative de rassembler les vues et présenter quelques meilleures idées de la pratique, afin de mieux contrôler l'activité de test. La norme a été révisée et mise à jour en 1998. Elle décrit huit documents qui peuvent être divisés en trois catégories selon (Copeland, 2004): les documents de préparation des tests, produits avant le développement du logiciel, les documents d'exécution des tests et les

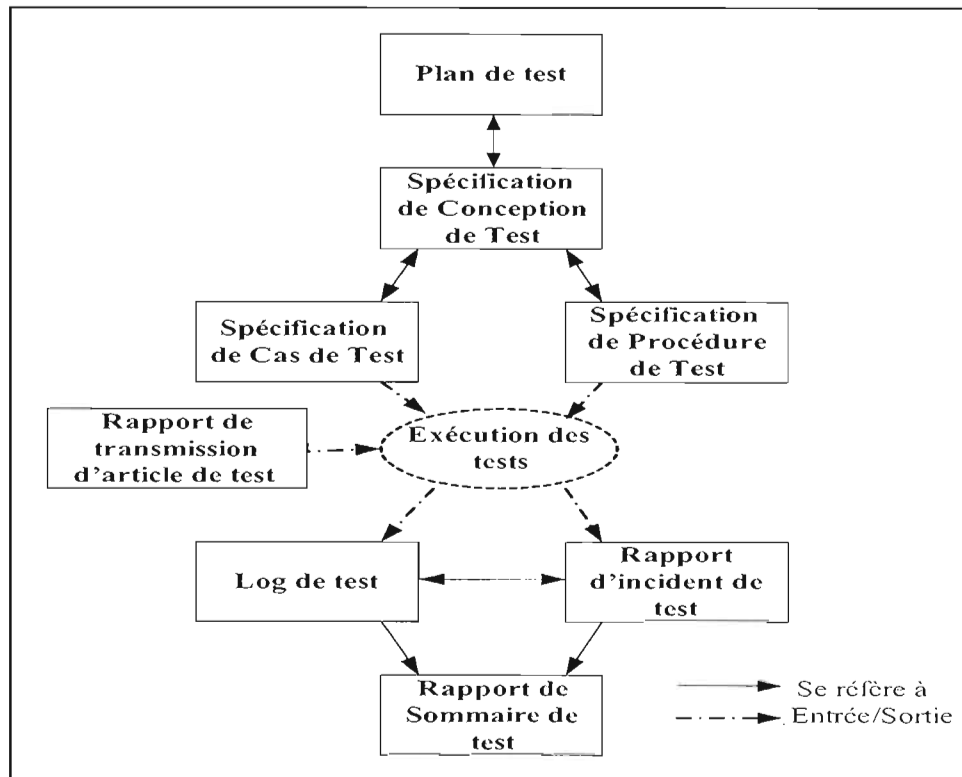
documents de complétude des tests. Chacune de ces catégories est composée de documents suivants:

- **Documents de préparation de tests**
 - Plan de test
 - Spécification de conception de tests
 - Spécification de cas de tests
 - Spécification de procédure de tests
- **Documents d'exécution de tests**
 - Rapport de transmission d'article de test
 - log (registre) de test
- **Documents de complétude de test**
 - Rapport d'incident de test
 - Rapport de sommaire de tests

La norme IEEE 829 (1998) a cité quelques avantages de son utilisation :

- L'utilisation des documents de tests standardisés pourraient faciliter la communication entre les intervenants de projet du développement en fournissant un protocole de référence commun;
- Le standard IEEE 829 pourrait servir comme un outil de vérification et d'évaluation (Check List) au processus de test;
- Un ensemble de documents normalisés selon cette norme pourrait également fournir une base pour l'évaluation des pratiques de documentation de test du logiciel;
- L'utilisation des documents selon la norme IEEE 829 permettrait de contrôler le processus de test. Cela résulte de l'augmentation de la visibilité de chaque phase du processus .

Figure 2.4 : Schématisation des documents de préparation de tests (Adapté et traduit de Copeland, 2004, p.189)



Dans notre étude, nous nous intéresserons seulement aux documents de préparation de tests, du fait qu'ils constituent le principal point de divergence entre le TS et le TE. La figure 2.4 montre les documents devraient être créés avant l'exécution de tests. Souvent ces documents sont créés parallèlement à l'implémentation du logiciel dans les vues préventives de test (Craig et Jaskiel, 2002). C'est l'une des forces de TS qui pourrait s'introduire très tôt dans le cycle de vie du logiciel et prévenir les erreurs et les ambiguïtés pendant la phase de spécification des exigences et la conception avant qu'ils se transforment à des défauts dans le code.

Notons que les documents du test sont également sujets à une validation. Cela peut être réalisé par une révision formelle à ces documents. À cet effet l'utilisation de la norme pourra faciliter énormément la mise en place de cette validation (Craig et Jaskiel, 2002). Cependant

selon Bach et al. (2002), la norme pourrait nuire à la qualité de test effectué. Du fait que les testeurs consacrent le temps limité du test à la création d'une documentation lourde et non nécessaire au lieu de l'investir dans le test du logiciel.

Selon (Swebok, 2004; Pyhajarvi et al., 2003; Craig et Jaskiel, 2002 ; Perry, 2000) le processus de test comporte les étapes suivantes: la planification, l'analyse et la conception de tests, l'exécution et l'évaluation de tests. Nous allons aborder dans les sections qui suivent chacune de ces étapes en mettant l'accent sur les documents créés et les éléments fondamentaux spécifiés dans ces documents.

2.5.1 La planification

La planification de l'activité de test peut commencer au moment de la formulation des besoins, se développer et se raffiner pendant la phase de conception du logiciel. Ce processus de planification donne naissance à un plan de test qui décrit les items (composants) et les caractéristiques de qualité (fonctionnalité, performance, utilisabilité, etc.) qui devraient être testés, ainsi que les responsabilités, les livrables et les besoins environnementaux requis et l'échéancier de projet de test. Sachant que ce plan de test peut être créé au niveau de projet (Master Test Plan) ou au niveau secondaire (unitaire, intégration, système, acceptation, etc.). Cela dépend de la complexité de logiciel et de l'organisation de projet. Il faut noter que cette planification est une activité continue, s'effectue tout au long du cycle de développement. Alors, les résultats de l'activité de test sont utilisés pour mesurer les risques et modifier le plan si nécessaire.

Le patron du plan de test de la norme IEEE 829 définit 16 clauses décrites ci-dessous, la description détaillée de chaque clause est présentée dans l'annexe B:

1. Identificateur du plan de test
2. Introduction
3. Items de test
4. Caractéristiques à tester
5. Caractéristiques qui ne devraient pas être testées
6. Approche de test

7. Critère de passage/échec
8. Critère de suspension et conditions de reprise
9. Livrables du test
10. Tâches du test
11. Besoins environnementaux
12. Responsabilités
13. Besoins en personnel et formation
14. Cédule
15. Risques et contingences
16. Acceptations

Le patron du plan de test présenté ci dessus fournit un croquis ou une structure de base du plan de test qu'il peut être adapté selon les besoins de chaque organisation. Pratiquement, la plupart des plans proposés dans la littérature divisent la clause risque et contingence en deux clauses séparées: la première décrit les risques liés au logiciel, et la deuxième les risques liés au projet de test et ses contingences (Craig, et Jaskiel, 2002). En effet, la rapidité suivie dans le développement et l'impossibilité de tester le logiciel exhaustivement ont poussé les intervenants dans le domaine du test à utiliser les risques du logiciel pour focaliser la stratégie et identifier la priorité des items de test. Il s'agit de faire une analyse de risques au début de projet de test en se basant sur les spécifications d'exigences (Craig, Jaskiel, 2002). Elle permet aussi de déterminer les zones à risques et les parties potentielles qui auraient tendance à avoir plus d'erreurs, et qui devraient être testées rigoureusement. Selon ces mêmes auteurs les résultats de cette analyse doivent être revus occasionnellement pendant le projet de test, du fait que les spécifications, les ressources, la portée de test et d'autres facteurs dans le projet peuvent se changer. D'ailleurs, le risque des composants qui ont subi des changements devient naturellement plus grand à chaque version. Cette analyse de risques pourrait servir aussi dans la mise en place de contingences convenables lorsqu'un événement inattendu survient et empêche l'exécution normale du plan de test.

2.5.2 Analyse et conception

La conception de tests est la première étape de développement de tests. Le processus

d'analyse et de conception de tests se consiste de trois étapes , à savoir: concevoir les tests en identifiant les conditions de tests, spécifier les cas de tests et spécifier les procédures de tests. Alors, pendant l'analyse, la documentation de base disponible dans cette étape, tels que les documents de spécification des exigences et de conception doivent être analysés soigneusement pour déterminer les items ou les articles qui devraient être testés. Une condition de test est définie comme un article ou un événement qui peut être vérifié par un ou plusieurs cas de tests, tel que une fonction ou caractéristique de qualité.

Pendant la spécification de cas de tests, les données de tests sont développées et décrites en détail en utilisant une ou plusieurs techniques de conception de tests (Beizer, 1995; Craig, Jaskiel, 2002). Le choix d'une technique dépend de la nature du système à tester, les objectifs visés et le risque global de l'exécution (Craig, Jaskiel, 2002). Cette phase se conclut par la production de trois livrables, *Spécification de Conception de Test*, *Spécification de Cas de Test* et *Spécification de Procédure de Test*. Elle se conclut aussi par l'élaboration de la matrice de traçabilité qui trace les exigences vers les cas de tests (Craig, Jaskiel, 2002).

Tableau 2.1 : La matrice de traçabilité

	Cas de test 1	Cas de test 2	Cas de test 3	Cas de test 4
Exigence 1	X	X	X	X
Exigence 2		X		
Exigence 3	X	X	X	
Exigence 4		X	X	
Totale	2	4	3	1

La matrice de traçabilité permet de tracer les cas de tests sur les exigences. Cela fournit un moyen pour identifier les exigences qui ne sont pas bien testées. Autrement, c'est un outil pour mesurer la couverture de tests (sera évoqué en détail dans le chapitre VII).

Cette matrice permettait aussi d'analyser l'impact de changements sur les exigences, et donne une idée du volume de travail nécessaire pour mettre à jour les cas de tests déjà conçus (Bach et al., 2002).

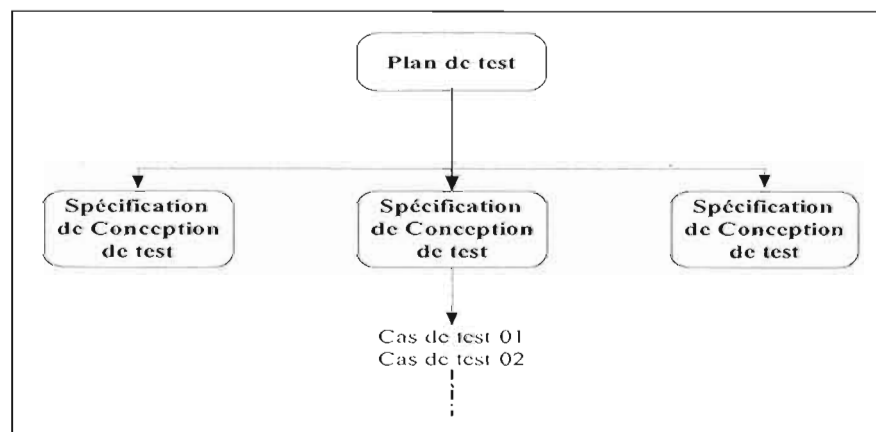
2.5.2.1 Spécification de Conception de test

Spécification de Conception de Test est un document spécifiant les conditions de tests (éléments de couverture) pour un article de test, l'approche détaillée du test et l'identification des cas de tests de haut niveau associés (IEEE 829, 1998). Il comporte les éléments suivants:

1. Identificateur de *Spécification de Conception de Test* (un nom unique pour distinguer le document parmi tous les autres).
2. Caractéristiques à tester (identifie les items et les caractéristiques objets de cette *Spécification de Conception de test*
3. Raffinements de l'approche (identifie les détails de la technique de test et de l'approche proposée dans le plan de test)
4. Identification de tests (fournit un identificateur unique et une courte description des cas de tests associés à cette *Spécification de Conception de test*)
5. Critère de succès/échec de test (critère utilisé pour déterminer si une caractéristiques passe où échoue le test)

En fait, le document de *Spécification de Conception de Test* est une miniature de plan de test. Son rôle est de regrouper les cas de tests semblables destinés à tester une ou plusieurs caractéristiques du logiciel, tel qu'illustré sur la figure 2.5.

Figure 2.5 : Spécification de Cas de Test



Par la suite les spécifications de chaque cas de test spécifié dans le document *Spécification de Conception de Test* devraient être déterminées.

2.5.2.2 Spécification de Cas de Test

Le but de *Spécification de Cas de Test* est d'identifier les détails de chaque cas de test cité dans la *Spécification de Conception de Test*. Le patron IEEE 829 de *Spécification de Cas de Test* décrit les éléments suivants :

1. Identificateur de Cas de Test (un nom unique qui distingue ce cas de test parmi tous les autres)
2. Items de test (items et caractéristiques objets du cas de test)
3. Spécification des entrées (spécifie les entrées requises par ce cas de test)
4. Spécification des sorties (spécifie les résultats prévus de l'exécution de cas de test)
5. Besoins environnementaux (matériel spécial ou logiciel nécessaire pour exécuter ce cas de test)
6. Exigences procédurales spéciales (identifie n'importe quelle procédure spéciale nécessaire pour installer l'environnement de test).
7. Dépendances inter-cas (cite les cas de test qui doivent être exécutés avant ce cas de test)

Comme nous venons de le voir, les résultats attendus devraient faire partie de Spécification de Cas de Test. Ces résultats constituent l'oracle de test dans le TS.

Selon Craig et Jaskiel (2002) l'approche IEEE 829 requiert une documentation complète de chaque cas de test, c'est la raison pour laquelle elle est utilisée dans les systèmes à grand risque. Selon ces mêmes auteurs, le patron ne constitue pas un bon choix pour tester les systèmes dynamiques et instables. En effet, la norme de documentation IEEE 829 demande un effort important dans la création des cas de tests, et quelques changements introduits sur le logiciel peuvent rendre ces cas de tests invalides.

Par contre, ce patron constitue un bon choix dans les organisations dynamiques qui se caractérisent par le changement fréquent de leur personnel ou qui ont du personnel inexpérimenté.

Par la suite, les cas de test conçus se mettent dans un ordre exécutable dans le troisième document de standard de documentation IEEE 829 de phase de conception : *la Spécification de Procédure de test*. Ces procédures de tests sont développées à partir des documents de *Spécification de Conception de Test* et *Spécification de Cas de Test*. Le document de *Spécification de Procédure de test* décrit comment le testeur junior devra exécuter physiquement le test. Une *Spécification de Procédure de Test* pourrait inclure plus qu'un seul cas de test.

2.5.2.3 Spécification de Procédure de Test

La *Spécification de Procédure de Test* est un document spécifiant la séquence d'actions pour l'exécution d'un test. Ce document connu aussi sous le terme script de tests ou script de tests manuels (IEEE 829). La norme définit dix étapes qui peuvent être utilisées dans l'exécution de tests, qui sont décrites ci-dessous :

1. Objet de la procédure
2. Exigences spéciales
3. Étapes de la procédure
 - *Log : comment enregistrer les résultats*
 - *Set Up : comment préparer l'exécution de la procédure*
 - *Start : comment débiter l'exécution de la procédure*
 - *Proceed : actions de la procédure*
 - *Measure : comment les mesures seront effectuées*
 - *Shut Down : comment suspendre la procédure de test*
 - *Restart : comment redémarrer la procédure de test*
 - *Stop : comment effectuer un arrêt ordonné de l'exécution*
 - *Wrap Up : comment restaurer l'environnement*
 - *Contingencies : comment traiter les anomalies durant l'exécution*

Nous pouvons constater de la description ci-dessus que le script de la procédure décrit en détail les étapes d'exécution de test et ne laisse rien à la volonté du testeur qui exécute le test. C'est l'une des critiques proposés par les membres de CDS contre le TS. Selon eux ce script transforme le testeur en robot exécutant les tâches sans aucune réflexion critique.

2.5.3 Exécution et évaluation

L'exécution des tests conçus se fait selon le planning décrit dans le plan de test. Pendant l'exécution des tests, le testeur junior enregistre les résultats de l'exécution dans les documents proposés par IEEE 829 : *Registre de test*, *Rapport d'incident de test*. Les défauts sont enregistrés dans un système de suivi des bogues. À la fin de l'activité de test, le document de standard IEEE 829 *Rapport de synthèse de Test* synthétise les activités et les résultats de test de la version testée du logiciel.

2.6 Conclusion

Nous avons donné dans ce chapitre un aperçu global sur la plupart des aspects touchant au TS en mettant l'accent sur les principales étapes du processus de TS. Il apparaît que le processus scénarisé est visible, planifié, incluant plusieurs métriques pouvant faciliter la mesure de l'efficacité de l'activité de test. Mais, dans la pratique le processus de test du logiciel ne se passe pas toujours de la même façon que dans la théorie, spécifiquement le test des systèmes dynamiques et changeants. S'ajoute à ce problème le fait que les testeurs n'interviennent qu'à la fin de la chaîne de développement dans les modèles traditionnels. Par conséquent l'activité de test s'effectuera souvent sous des pressions du temps, de coût et le besoin de livrer le logiciel. À cet effet, les compagnies de développement de logiciels ont commencé à chercher des méthodes pouvant mieux s'adapter avec ces réalités. Le TE constitue une de ces méthodes, il fera le sujet de prochain chapitre.

CHAPITRE III

TEST EXPLORATOIRE : VUE D'ENSEMBLE

Ce chapitre propose une vue d'ensemble du test exploratoire, en présentant brièvement les aspects fondamentaux de cette approche. Nous commençons par la définition de test exploratoire (TE), puis l'accent sera mis sur l'approche de test «Session Based Test Management» (SBTM) et ses mécanismes principaux. Enfin, quelques techniques et styles d'exploration seront décrits et nous terminerons par une conclusion.

3.1 Définitions

Au début des années 90, les membres de Context Driven School (CDS) ont commencé à utiliser le terme «exploratoire» pour décrire cette nouvelle pratique de test. Cette terminologie a été publiée d'abord par Kaner (1988):

*«...() At some point, you'll stop formally planning and documenting new tests until the next test cycle. You can keep testing. Run new tests as you think of them, without spending much time preparing or explaining the tests. Trust your instincts... In this example, you quickly reached the switch point from formal to informal testing because the program crashed so soon. Something may be fundamentally wrong. If so, the program will be redesigned. Creating new test series now is risky. They may become obsolete with the next version of the program. Rather than gambling away the planning time, try some **exploratory tests** whatever comes to mind. »* d'après (Kaner, 1988)

Comme nous pouvons le constater, l'auteur a défini le TE comme la conception et l'exécution de tests à temps réel, dès qu'une nouvelle idée de test s'émerge, sans perdre du temps dans la planification et la préparation de tests formels qui pourront devenir obsolètes à la prochaine version vue l'instabilité du système.

À l'occasion du 7ème atelier de Los Altos sur le test du logiciel, les praticiens et les chercheurs participants ont tous collaboré pour définir le TE. Ils ont accentué certaines des caractéristiques communes de leurs vues et se sont mis d'accord sur les caractéristiques de TE citées ci dessous (Kaner, Tinkham, 2003a):

- Interactif ;
- Combinaison de la cognition et l'exécution;
- Créatif ;
- Mène à des résultats rapides;
- Diminue l'archivage des documents de test.

En 2002, dans le premier livre qui présente les idées et les principes de la pensée «test piloté par le contexte» CDS, Bach et al.(2002) ont défini le terme exploration comme une interrogation déterminée, c'est à dire une navigation avec une mission générale sans itinéraire scénarisé.

«By exploration we mean purposeful wandering: navigating through a space with a general mission, but without a pre-scripted route. Exploration involves continuous learning and experimenting... () » d'après (Bach et al., 2002)

Ces mêmes auteurs ont présenté le TE comme une technique de test où le testeur apprend le produit logiciel, son marché, ses risques et ses défauts antérieurs tout au long de l'activité de test pour concevoir des nouveaux tests plus puissants grâce à l'évolution et à la maturité de ses connaissances (Bach et al., 2002).

Kaner et Tinkham (2003a) ont défini le TE comme n'importe quelle activité de test, dans la mesure où le testeur contrôle activement la conception des tests. Pendant que ces tests s'exécutent, il utilise l'information résultante pour concevoir de nouveaux tests. Par cette proposition, les auteurs tendent à généraliser la définition au test scénarisé (TS) qui pourrait être exécuté dans certaines situations d'une façon exploratoire comme nous allons le voir dans les lignes qui suivent.

Cependant, la définition la plus fréquente dans la littérature est celle donnée par Bach (2003).

Il définit le TE comme l'apprentissage, la conception et l'exécution simultanée des tests.

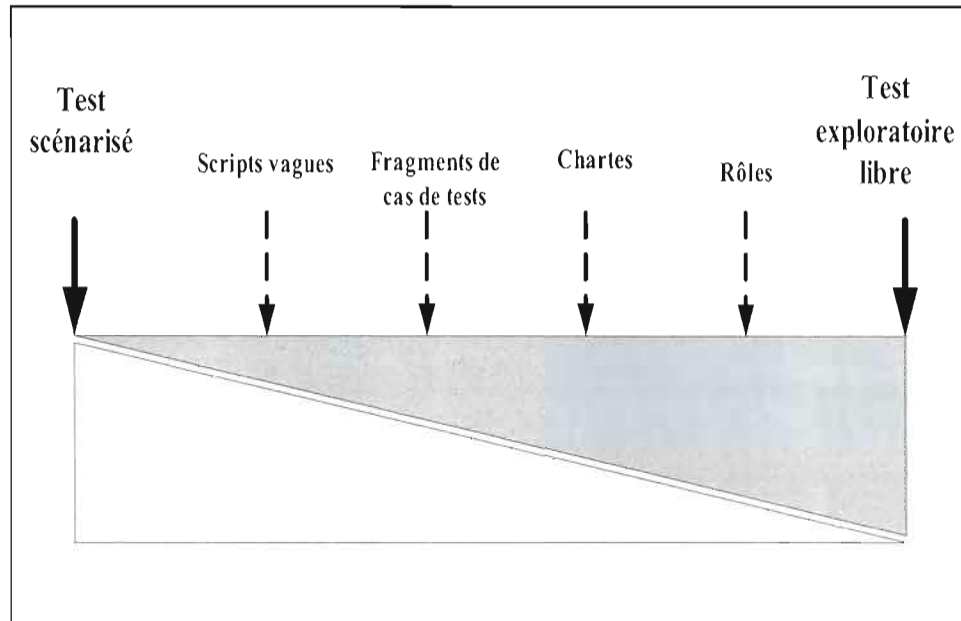
Le TE a été reconnu par le guide Swebok (2004) comme une technique valide de test. Cependant, il relie l'efficacité de TE à la connaissance de l'ingénieur logiciel ou le testeur.

D'après les définitions ci-dessus, nous pouvons déduire les propriétés maîtresses de TE:

- L'apprentissage, l'exploration, la conception et l'exécution des tests se font simultanément. Autrement dit, les tests ne sont pas définis à l'avance comme les cas de tests scénarisés.
- L'activité de test est guidée à chaque moment par les résultats des tests antérieurs dans une boucle interactive d'appréhension du logiciel sous test.
- Le TE n'exige pas la disponibilité des exigences du logiciel, du fait qu'il se fonde sur l'exploration et l'apprentissage pendant le test.
- Centrée autour de la détection des défauts, c'est-à-dire orienté vers le résultat plutôt que la préparation, la documentation et l'archivage des cas de tests.

Nous pouvons constater de ces propriétés qu'il y a une différence claire entre le TE et le TS. La réalité des pratiques de test démontre que cette différence est inaperçue, du fait que même une activité de TS pourrait être exécutée d'une façon exploratoire. Un exemple clair de telle situation apparaît quand le testeur dévie de ses procédures de tests et utilise l'exploration pour explorer un défaut pendant l'exécution de ses tests scénarisés (Kaner, Tinkham, 2003a). Nous pouvons donc conclure que n'importe quelle activité de test logiciel réalisée par un être humain pourrait être *exploratoire* à un certain degré. Selon Bach (2003) n'importe quel effort de test tombe quelque part sur un continuum (figure 3.1) dont un des pôles représente le TS, où le testeur suit exactement les procédures de tests scénarisés dans chaque détail et l'autre pôle représente le TE le plus libre (Freestyle Exploratory Testing) où les idées de test émergent au moment de leur exécution.

Figure 3.1 : Continuum repérant l'activité de test (Adapté et traduit de Bach, 2007)




La figure 3.1 situe plusieurs variantes de test entre les deux paradigmes, le TE et le TS. Chacune de ces variantes accentue un niveau de formalisme, de détail, de documentation et de contrôle par rapport au TE libre. Elle pourrait prendre la forme des rôles ou des responsabilités pour chaque membre de l'équipe de test sur une partie du logiciel. Elle pourrait prendre aussi la forme des chartes qui sont plus précises que les rôles. Ces chartes identifient la durée de la mission avec une liste précise des items à tester. Les deux derniers types sont les deux modèles de TE les plus proches de TS. Tout d'abord les fragments de cas de test qui pourraient spécifier les mêmes éléments que *Spécification de Cas de Test* dans la norme IEEE 829, sans spécifier toutefois les détails fins comme le cas de test scénarisé. Quant aux scripts vagues, ils sont plus précis que les fragments de cas de test et similaires aux procédures. Ils contiennent les étapes à suivre pour accomplir la mission de test, mais sont moins détaillées que celles-ci et nécessitent de l'exploration pour les accomplir. Avant de fermer cette parenthèse, notons que certaines organisations ont utilisé les patrons IEEE 829, spécialement, les spécifications des cas de tests pour intégrer le TE dans leurs pratiques de test, avant l'introduction du concept de «Session Based Test Management » (Amland et Vaga, 2002).

3.2 Processus de test exploratoire


Le processus de TE est un processus tridimensionnel. Tel qu'illustré sur le tableau 3.1, ces dimensions sont: produit, qualité et techniques (Bach, 2003). Selon l'auteur, un testeur exploratoire teste un *produit logiciel*, évalue sa *qualité*, en utilisant diverses *techniques*. Chaque case dans la grille ci-dessous représente un aspect du processus de TE. Le testeur exploratoire peut choisir n'importe quel point de départ et suivre n'importe quel chemin dans la grille jusqu'à la fin de la session de test, à condition que les neuf tâches soient couvertes pendant le cycle de test et que chacune des trois activités principales : apprentissage, conception de tests et exécution de tests donnent un résultat tangible.

Tableau 3.1 : Grille des tâches de test exploratoire (Adapté et traduit d'Amland, 2002a)


Grille des tâches	Apprentissage	Conception de tests	Exécution de tests
Produit (couverture)	Découvrir les éléments du produit	Décider quoi tester	Observer le comportement du produit
Qualité (Oracle)	Découvrir comment le produit devrait fonctionner	Penser aux problèmes de qualité possibles	Évaluer le comportement contre les prévisions
Techniques	Découvrir les techniques de conception des tests qui peuvent être utilisées	Choisir et appliquer les techniques de conception de tests	Configurer et exercer le produit.



Les notes de tests



Les tests



Les problèmes trouvés

Selon Bach (2001) le TE peut être pratiqué selon un plan prévu qui n'est pas nécessairement rigoureux. Du fait qu'un plan rigoureux exige la certitude et implique la perfection. Celles-ci ne pourraient pas être atteintes dans une activité de TE qui se fonde sur l'exploration et l'apprentissage du logiciel pour identifier ce qui doit être testé. Cependant Bach signale

qu'un bon TE est une activité planifiée, et la préparation de quelques notes sur la stratégie à suivre et les éléments du logiciel à aborder dans le test pourraient être très utiles.

En ce qui concerne les types de TE, deux types ont été traités dans la littérature : le premier, est le test exploratoire libre (Freestyle Exploratory Testing). C'est un test qui se fait sur des intervalles limités de temps qu'on appelle *sessions*, chacune munie d'une charte. La charte décrit la mission de test et parfois identifie quelques tactiques et techniques de test pouvant être utilisées pour exécuter cette mission. La charte pourrait être choisie par le testeur ou assignée par le responsable du test. Les résultats officiels de ce type sont constitués seulement des bogues détectés pendant la session de test (Bach, 2003). Le deuxième type de TE est le test exploratoire géré par session (Session Based Test Management: SBTM). C'est une approche structurée de TE introduite par Jonathan Bach (2000) pour contrôler le TE libre. Dans ce type de TE les résultats officiels sont constitués des bogues détectés dans la session de test et un rapport de session qui fait l'objet d'une évaluation par le responsable de test. Les mécanismes et les métriques de ce type de TE vont être évoqués en détail dans la section qui suit.

3.3 Test exploratoire géré par session (SBTM)

L'apparition du TE dans sa version originale, c'est-à-dire tel que présenté au début, un processus libre et informel, a suscité beaucoup de critiques de la part des praticiens et des professionnels. Ces critiques étaient centrées sur le manque de contrôle et de mesure qui sont importants et cruciaux dans l'industrie de test de logiciel. En effet, le fait que le TE ne soit pas scénarisé, ni répétitif et qu'il dépend fortement de la créativité du testeur, a rendu difficile le contrôle et le suivi de la progression de projet de test. Compte tenu de ces faiblesses, les intervenants dans le domaine de test ont trouvé difficile d'essayer ou d'introduire le TE tel que présenté la première fois comme une méthodologie de test.

Pour pallier ces problèmes, Jonathan Bach (2000) a proposé une nouvelle approche de TE nommé *Test géré par session* (Session Based Test Management (SBTM)). Le but de l'approche est de rendre le TE plus responsable (Accountable) et d'introduire la mesure et le contrôle nécessaires pour la gestion de projet de test. L'idée principale de la méthode SBTM est de planifier et diviser la mission générale de test en plusieurs sessions. Une session est un

intervalle de temps ininterrompu qui pourrait durer de 60 à 120 minutes (Bach.J, 2000).Chaque session est identifiée par une charte qui décrit la mission de test à remplir. Nous pouvons dire que c'est un plan de haut niveau, son rôle est de spécifier les tâches de test à remplir, ainsi que quelques tactiques et techniques pour les exécuter. Notons que la granularité de détails de chaque charte varie selon l'importance de celle-ci en termes de risque et de la difficulté de la mission.

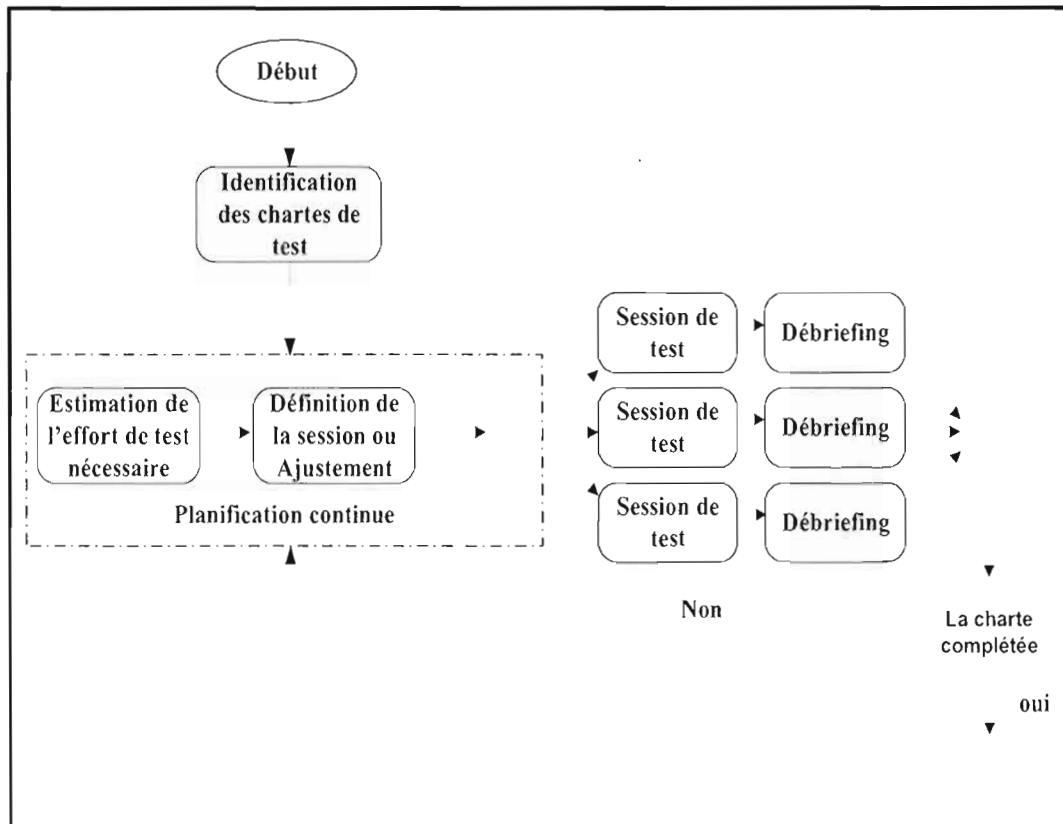
Selon Jonathan Bach (2000) chaque session devrait être divisée en trois étapes qui comportent autant de métriques pour contrôler la portée de travail du testeur. Ces métriques sont:

- ***Préparation de test:*** le pourcentage du temps de la session consacré à l'installation de l'environnement de test (configurer matériels de test, lire quelques manuels, etc.).
- ***Conception et exécution de tests:*** le pourcentage du temps de la session passé dans l'exploration et le test.
- ***Investigation et rapport de bogues:*** le pourcentage du temps de la session passé dans la recherche et l'investigation lors de l'apparition d'un comportement inattendu du logiciel. Plus, le temps passé sur le rapport des bogues.

Le testeur devrait rapporter aussi l'estimation du temps passé sur la charte par rapport au temps passé sur « l'opportunité ». L'opportunité c'est le test effectué par le testeur qui n'est pas inclus dans la charte de session, puisque le rôle de l'approche SBTM selon Jonathan Bach (2000) est d'introduire le contrôle et la mesure sur le TE libre tout en gardant ses aspects essentiels que soient l'improvisation, l'exploration et la créativité.

De plus, le rapport de session devrait rapporter les bogues, les problèmes et les notes. Les problèmes sont les questions et les obstacles reliés au processus du test. Quant aux notes, elles présentent des idées et des pistes qui peuvent amener à la création de nouvelles chartes de test. Le rapport de chaque session fait alors l'objet d'une évaluation pendant le débriefing avec le responsable du test.

Figure 3.2 : Cadre d'application de SBTM (Inspiré de James et Wood, 2003)



Tel qu'illustré à la figure 3.2, au début de l'activité du test, le responsable du test identifie les chartes de tests et l'effort nécessaire pour accomplir chacune d'elles. Après l'exécution de chacune de ces chartes, le responsable rencontre le testeur pour évaluer son travail. Suite à ce débriefing, le responsable décidera si l'exécution de charte est complétée ou si la session devrait être ajustée et prolongée pour compléter le test de charte. Le responsable du test pourrait aussi ajouter de nouvelles chartes pour explorer les notes rapportées par le testeur. De ce fait, le processus d'identification de chartes est un processus de planification continue (Wood et James, 2003) se change régulièrement pendant l'activité de test au fur et à mesure que des nouvelles informations apparaissent pendant l'exécution des chartes. Les résultats de sessions de tests sont la plupart du temps rassemblés dans une base de données. Ainsi, le pourcentage de sessions complétées, les bogues rapportés et le rendement des testeurs

pourraient être retracées par les responsables du test. Les renseignements sur la progression et le statut de l'activité de test pourraient être disponibles à chaque moment de projet. En effet, les mesures collectées pendant le test et le débriefing permettent d'estimer la productivité ou le rendement de chaque membre de l'équipe de test pendant le projet de test en cours. Cela avec le nombre de sessions complétées, pourrait aider dans l'estimation de quantité du travail restante avant la fin du cycle de test.

Une expérience d'utilisation de cette approche a été présentée par (Lyndsay et van Eeden, 2003). Les auteurs ont proposé des méthodes pour contrôler la portée de test et évaluer et mesurer la couverture de test. Les résultats de cette expérience seront abordés en détail dans le chapitre IV.

3.4 Les styles et les techniques d'exploration

Depuis l'apparition de TE, plusieurs praticiens et chercheurs se penchaient sur l'élaboration des techniques et des modèles d'exploration (Bach et Jonathan Bach, 2006; Bach et Kaner, 2004; Amland, 2002b). Dans cette perspective, Amland (2002b) ont proposé quelques styles et techniques pour effectuer le TE. Ils incluent entre autres :

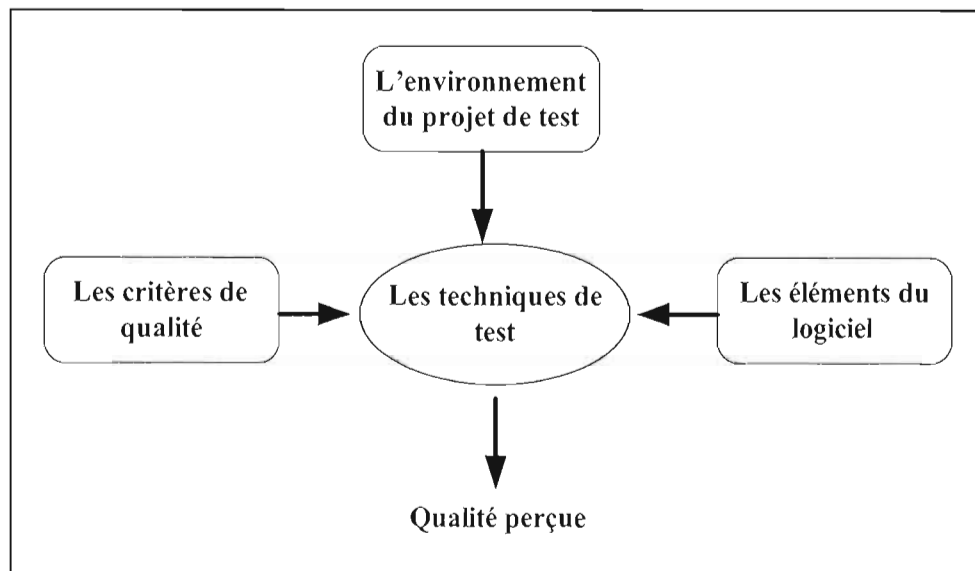
- Les intuitions: sont les idées que le testeur pourrait générer en se basant sur ses prédictions, ses compétences et son expertise.
- Les modèles: il s'agit de certains diagrammes et modèles qui peuvent aider le testeur dans l'appréhension du logiciel et la conception de tests. Ils incluent entre autres :
 - Diagramme d'architecture: consiste à construire ou concevoir un diagramme d'architecture du logiciel sous test, incluant ses interfaces, son flux de données et ainsi de suite. En pratique ce travail se fait la plupart du temps pendant des réunions de groupe, entre les testeurs et les programmeurs. Souvent l'identification de chartes de tests et les responsabilités se fait à cette étape où chaque testeur détermine sa mission convenablement aux parties du logiciel comprises.

- ❑ Digrammes d'états : consiste à créer un diagramme représentant les modes opérationnelles, les actions et les entrées possibles du logiciel. Selon Amland (2002b) ce digramme est un outil utile pour exposer les contradictions du logiciel.
 - ❑ Modèles de défaillances: consiste à utiliser un catalogue de bogues typiques pour concevoir les cas de tests. Le testeur exploratoire peut utiliser ses expériences antérieures pour construire ce catalogue ou utiliser en un de la littérature (Kaner et al., 1999)
- Exemples : il s'agit de certains exemples d'utilisation du système qui peuvent indiquer les anomalies et les défauts existants. Ils incluent entre autres :
 - ❑ Les cas d'utilisations : il s'agit de déterminer les utilisateurs du système et les tâches fonctionnelles pour chacun d'entre eux, ensuite on crée des tests qui reflètent leurs utilisations.
 - ❑ Opéra savon : il s'agit de générer des scénarios d'utilisations du système sous test en exagérant dans quelques-unes de ses aspects, par exemple en utilisant des valeurs extrêmes (limites) pour le même scénario.
- Les interférences : il s'agit de certaines actions qui peuvent nuire à l'exécution normale du système, comme des arrêts subits, la concurrence avec d'autres systèmes, etc. Ces interférences peuvent révéler des défauts dans le logiciel.
- Gestion d'erreurs : il s'agit de vérifier que le logiciel gère bien les erreurs d'utilisation, autrement dit, de vérifier que les messages d'erreurs se déclenchent au bon moment et sous les bonnes conditions.

Il faut rappeler que le questionnement constitue le cœur de TE. Il constitue la base de n'importe quelle technique ou style d'exploration. La qualité du TE effectué dépend de la qualité des questions générées à propos du produit sous test (Bach, 2003; Kaner, Tinkham,

2003b; Kaner, Amland, 2002b). Dans le but de faciliter la tâche du testeur exploratoire dans l'élaboration des questions et des stratégies efficaces, quelques praticiens et chercheurs ont proposé quelques modèles, comme le modèle d'analyse proposé par Bach (1996), qui est illustré à la figure 3.3, et les modèles d'attaques du logiciel proposés par Whittaker (2003).

Figure 3.3 : Heuristic Test Strategy Model (Adapté et traduit de Bach, 1996)



Ce modèle présente quatre secteurs principaux, chacun étant un indicateur que le testeur peut utiliser pour déterminer l'information dont il a besoin pour concevoir une stratégie de test. L'objectif immédiat de ce modèle est donc de guider la réflexion des testeurs exploratoires lorsqu'ils créent les tests. Ses principaux composants sont:

- L'environnement de projet: inclut les ressources, les contraintes, et d'autres facteurs qui peuvent aider ou nuire au processus de conception et d'exécution des tests (client, calendrier, équipement, etc.).
- Les éléments du produit sont les aspects visibles ou invisibles du produit, comme les structures de données, les interfaces, etc.

- Les critères de qualité sont les normes et les caractéristiques de qualité que le logiciel devrait respecter. Ces critères permettent au testeur de déterminer les problèmes dans le logiciel sous test.
- Les techniques de tests sont les stratégies nécessaires pour concevoir les tests. Le choix des techniques de test dépend de l'environnement de projet, les éléments du produit sous test et les critères de qualité visés dans le projet de test en cours.
- La qualité perçue est le résultat prévu du test.

L'environnement de projet, les critères de qualité et les éléments du produit sont tous combinés avec les techniques de test afin de déterminer la qualité perçue du produit. Selon Kaner et Bach (2004) le modèle aide le testeur exploratoire à déterminer ce qui est doit être testé. Ainsi, les attributs de qualité les plus importants dans le projet (les types de problèmes à chercher pendant l'activité de test), les aspects de projet qui pourraient faciliter ou contrarier l'activité de test en cours. La réflexion selon ces trois axes, pourrait générer des idées de test intéressantes qui peuvent être mises en œuvre en respectant les contraintes et les ressources du projet.

Nous croyons que les secteurs décrits dans ce modèle d'analyse sont semblables et ont les mêmes utilités que les secteurs identifiés dans le plan de test IEEE 829 (Annexe B).

Cependant, le choix d'un style spécifique d'exploration ou tactique particulière de TE dépend selon Kaner et Tinkham (2003b) de facteurs décrits ci-dessous:

- Les expériences antérieures;
- Les qualifications ;
- La personnalité, surtout le modèle d'apprentissage ;
- Les connaissances antérieures sur l'application sous test.

Selon ces mêmes auteurs, le facteur le plus pertinent est le modèle d'apprentissage, qui représente la façon dont la personne choisit et traite l'information (Kaner, Tinkham, 2003b). Cependant, cette hypothèse est théorique et n'est pas toujours confirmée par une étude empirique.

3.5 Conclusion

Nous avons donné, dans ce chapitre, un aperçu global sur la plupart des aspects touchant au TE, en commençant par sa définition et les concepts principaux du son processus. Puis nous avons présenté l'approche SBTM et ses mécanismes. En terminant nous avons proposé quelques techniques et modèles d'exploration. Tout le matériel dont nous avons discuté dans ce chapitre a été élaboré dans le but d'aider et encourager les testeurs et les organisations à adopter le TE. Mais, comment les entreprises vont adopter cette nouvelle approche, et quels sont les motifs et les raisons qui les ont poussées à essayer et utiliser le TE, en mettant à l'écart la pratique scénarisée habituelle. Nous allons proposer des réponses à toutes ces questions dans la revue de littérature de quelques travaux et études de cas dans le chapitre qui suit.

CHAPITRE IV

REVUE DE TRAVAUX RELIÉS

Dans ce chapitre nous allons présenter une revue des travaux de recherches académiques et professionnelles traitant du test exploratoire (TE). Dans la première partie, nous décrirons les résultats de la seule recherche académique existante à date. Elle met l'accent sur les raisons et les façons d'utilisation de TE et recense les bénéfices et les imperfections perçus par les praticiens dans l'industrie de test. Puis nous proposerons quelques expériences d'utilisations de TE qui ont fait l'objet de plusieurs conférences internationales. Notre but sera de définir et de comprendre comment et pourquoi les praticiens adoptent et adaptent le TE dans l'industrie de test. Cela va nous permettre d'identifier les facteurs influençant l'adoption et l'adaptation de l'approche dans l'industrie. Ces facteurs vont nous aider dans la construction de notre cadre comparatif qui va guider notre analyse comparative entre les deux approches, le test scénarisé (TS) et le test exploratoire (TE).

4.1 Étude de Itkonen et Rautiainen (2005)

La croissance remarquable d'utilisation de TE dans l'industrie de test logiciel et la promotion étendue de son efficacité par quelques praticiens, ont motivé les auteurs Itkonen et Rautiainen (2005) à étudier l'approche afin de dévoiler ses bénéfices annoncés. Ils ont retenu la question suivante: « pourquoi et comment les compagnies utilisent-elles le TE ? » pour aborder leur recherche. Dans le but de répondre à cette question, ils ont entrepris trois études de cas descriptives auprès de trois entreprises finlandaises. Une d'entre elles est petite, avec environ 10 employés travaillant dans le développement du logiciel et n'a qu'un seul produit sur le marché au moment de l'enquête. Sa méthodologie de test se fonde sur le TE due à l'immaturité de son processus de développement. Les deux autres entreprises sont moyennes, avec environ 30 et 40 employés dans le développement du logiciel. Ses produits

ont été sur le marché pendant plusieurs années. Leurs méthodes de test incluent les deux approches de test, le TE et le TS.

Itkonen et Rautiainen (2005) ont élaboré sept interviews thématiques, semi-structurées avec les praticiens effectuant le TE dans les trois entreprises. Ils ont interviewé successivement un seul testeur de la plus petite entreprise participante dans l'étude, qui avait utilisé le TE seulement deux fois avant l'entrevue. Puis quatre testeurs de la deuxième entreprise où le TE a été introduit dans la méthodologie de test six mois avant les entrevues. Enfin, deux testeurs de la plus grande entreprise dans l'enquête où le TE avait été utilisé et amélioré pendant quatre ans. Les auteurs ont utilisé des thèmes et des questions ouvertes et neutres, afin de recueillir les opinions et les expériences réelles des intervenants en mettant l'emphasis sur les raisons et les modes d'utilisation du TE dans les trois entreprises étudiées. En même temps, ils ont recensé les imperfections et les bénéfices du TE, tels que perçus par les praticiens. Parallèlement à leur étude descriptive, Itkonen et Rautiainen (2005) ont recueilli des données quantitatives sur le TE afin d'en mesurer la productivité.

4.1.1 Les raisons d'utilisation du test exploratoire

Les résultats de cette étude ont indiqué que le TE est utilisé pour les raisons à savoir :

- La difficulté de concevoir des cas de test scénarisés détaillés à cause de l'interdépendance entre les unités logiciel,
- Le besoin de tester le logiciel du point de vue de l'utilisateur final,
- Le besoin d'exploiter la créativité et l'expérience des testeurs dans la détection des défauts importants,
- Le besoin de fournir aux développeurs un feedback rapide sur les nouvelles unités logicielles,
- La capacité du TE de s'adapter aux contraintes des environnements dynamiques de développement et les situations où les exigences du logiciel manquent.

4.1.2 Les modes d'utilisation du test exploratoire

Les auteurs Itkonen et Rautiainen (2005) ont identifié, en se basant sur les informations recueillies auprès des interviewés, cinq modes d'utilisation principaux de TE dans les trois études de cas réalisées. Selon les constations de Itkonen et Rautiainen (2005) l'intuition a été utilisée comme technique de base dans le TE. Aucune autre stratégie ou technique spécifique d'exploration n'a été utilisée parmi celles proposées par (Kaner et Bach, 2004). Les auteurs ont identifié les modes d'utilisation suivants:

Test exploratoire basé sur session : deux des trois entreprises étudiées utilisent le TE géré par session connu sous le terme Session Based Exploratory Testing (SBET). Il consiste à utiliser le principe de la technique SBTM proposé dans le chapitre III, sans implémenter les métriques de gestion proposées par (Jonathan Bach, 2000).

Test fonctionnel d'une unité logicielle : Il s'agit de tester une unité logicielle individuelle directement après l'implémentation de celle-ci, pour produire un feedback rapide aux développeurs, très tôt dans le cycle de vie du logiciel.

Test exploratoire fumigatoire (Smoke Exploratory Testing) : Ce type de TE est utilisé par la plus grande entreprise participante dans l'étude. Il consiste à explorer chaque version du système à tester par l'équipe de test avant le début de test scénarisé, pour formuler une vue globale sur la qualité du système et s'assurer que les réparations sont proprement implémentées et que les fonctions les plus cruciales fonctionnent, sans se préoccuper des petits détails.

Test exploratoire de régression : deux des trois entreprises étudiées utilisent le TE dans le test de régression pour vérifier que les modifications n'ont pas causé d'effets inattendus à d'autres parties du logiciel. Il s'agit d'explorer le système dans une courte session, sans aucune planification. Les résultats de cette session sont informellement communiqués aux développeurs. En fait, la limitation de ressources et du temps ont été les motifs principaux pour utiliser le TE dans un test de régression, au lieu d'utiliser le test de régression habituel par une répétition sélective des cas de tests déjà conçus.

Test exploratoire libre : ce type de test est perçu par les interviewés dans la grande entreprise comme une pratique systématique et naturelle qui devrait se faire pendant l'exécution des cas de test scénarisés.

4.1.3 Les bénéfices du test exploratoire

En ce qui concerne les bénéfices perçus de TE, tous les interviewés ont illustré que la souplesse et la flexibilité de l'approche leur permettrait de tester en profondeur les unités logicielles qui ne pourront pas être traitées dans les cas de tests scénarisés, comme les interdépendances entre les anciennes et les nouvelles unités logicielles (Itkonen et Rautiainen, 2005). Selon les constatations de ces mêmes auteurs, la productivité en terme de nombre des défauts détectés et l'importance de ces défauts a été perçue comme un bénéfice. Cependant, les interviewés ont relié la productivité à l'expertise des testeurs dans le TE. Ils affirment que le TE ne pourra pas être productif si le testeur n'a pas des connaissances adéquates dans le domaine d'affaire du logiciel à tester. Ils ajoutent que la différence dans l'attitude pendant le test joue un rôle crucial dans la productivité perçue de TE. En effet, le testeur tend plus à vérifier l'exactitude entre les résultats observés et les résultats prévus identifiés dans les cas de test scénarisés dans une activité de TS. Par contre dans le TE, le testeur aborde le logiciel avec une attitude « offensive ». Autrement dit, il cherche les défaillances avec de la curiosité et un œil critique (Itkonen et Rautiainen, 2005). Les répondants ont affirmé aussi que le TE est productif seulement dans les courtes périodes de test en considérant le nombre d'heures utilisées et le nombre de défauts détectés. Tandis qu'à long terme il ne pourrait pas être productif à cause de la difficulté d'estimer la couverture de tests pendant l'activité de test. Par la suite, quelques parties ou caractéristiques du logiciel peuvent être livrées sans être testées (Itkonen et Rautiainen, 2005).

4.1.4 Les lacunes du test exploratoire

Selon les constatations de Itkonen et Rautiainen (2005) la couverture limitée est la plus grande lacune de TE.

Ceci a été mentionné par tous les interviewés dans l'enquête qu'ils ont entreprise.

Les résultats ont montré que la dépendance de TE à la créativité et à l'expérience des testeurs a été perçue aussi comme un désavantage du TE, du fait que le TE est sujet aux erreurs humaines.

Malgré la profondeur et la rigueur de cette recherche, elle est limitée par le nombre et la taille des entreprises participantes. Ceci apparaît clairement à plusieurs reprises dans la recherche où les résultats obtenus concernent une seule entreprise parmi les trois participantes. Donc, nous ne pouvons pas savoir si les résultats obtenus sont généraux ou des cas isolés. En ce qui concerne la productivité de l'approche dans la détection de défauts, la recherche n'a pas proposé des preuves fiables. En fait, les données quantitatives collectées ne peuvent pas être définitives. À cause de l'absence des données quantitatives du TS, qui pourraient constituer une base de comparaison. L'étude est quand même un apport utile à l'enrichissement de la littérature sur les justifications et les modes d'utilisation du TE.

4.2 Étude de Petty (2005)

Petty (2005) présente une expérience d'utilisation de l'approche Session Based Exploratory Testing (SBET) comme une méthodologie primaire de test en remplacement de la méthode scénarisée habituelle. Cette dernière se trouvait incapable de s'adapter aux changements fréquents des exigences du logiciel. Alors, l'introduction de la méthode SBET a été perçue comme une solution, vu les frustrations accumulées de l'utilisation de TS. Ces réalités résident dans le changement continu des exigences et l'absence du temps suffisant de test du logiciel. La méthode SBET adoptée par Petty (2005) est inspirée dans une grande partie de celle proposée par Jonathan Bach (2000). Cette méthode a permis aux testeurs d'être agiles dans le sens où ils ont été capables de s'adapter à l'incertitude et au changement de logiciel et de tester adéquatement le logiciel à un coût optimal.

Selon les constatations de Petty (2005) la chose la plus intéressante dans la méthode SBET est qu'elle a éliminé le besoin de retravailler et de mettre à jour les cas de tests scénarisés après chaque changement du logiciel. Ce temps, selon l'auteur, a pu être investi dans le test et l'amélioration de qualité du produit logiciel.

Petty (2005) a utilisé des pairs, c'est-à-dire que deux personnes s'assoient à seul ordinateur et exécutent la même mission de test. Chacune de ces paires est composée d'un testeur et d'un développeur. Selon les constatations de l'auteur, l'utilisation de l'approche SBET en pair a amélioré considérablement la qualité de test effectué. En effet, dans une session de test par pairs, un membre de la paire peut se concentrer sur la conception des tests et l'autre sur l'enregistrement et la rédaction de notes. Les rôles des membres de la paire sont échangés plusieurs fois pendant la session. Cela augmente la créativité et la concentration du membre qui conçoit les tests et élimine la distraction qui pourrait être causé par l'enregistrement des notes. Aussi, la collaboration et le partage des connaissances tacites des membres de l'équipe pendant la session ont amélioré la compréhension et l'apprentissage du logiciel sous test. Notons que l'utilisation de développeurs dans les pairs a permis de corriger les défauts en temps réel sans avoir besoin d'enregistrer beaucoup de notes de session ni de les reproduire ultérieurement.

Selon les constatations de Petty (2005), le fait que l'approche SBET soit basée sur l'exploration et l'apprentissage a poussé les testeurs à s'impliquer plus dans le processus de conception et de clarification des exigences, pour pouvoir comprendre le produit logiciel et son domaine d'affaire. Cela a eu des répercussions positives sur la qualité des tests effectués et sur la capacité de détecter des défauts ultérieurement pendant le test. Les testeurs sont devenus plus capables de différencier entre un défaut et un comportement normal du logiciel. Ce concept est connu sous le terme d'oracle de test. Il sera abordé plus en détail dans le chapitre VII.

Selon les constatations de Petty (2005), l'approche SBET a été très efficace dans le cas de leur projet de test. Ce projet se caractérisait par le changement fréquent des exigences, qui sont souvent communiquées verbalement. Par la suite la méthode SBET lui a permis de pouvoir répondre à ces changements rapidement. Selon les constatations de l'auteur le moral de l'équipe de test a augmenté considérablement pendant le test du logiciel. À cause de l'élimination du fardeau habituel de la mise à jour des cas de tests lors de changement du logiciel. La communication entre les testeurs et les développeurs s'est améliorée et transformée d'une relation d'adversité à une relation de collaboration. Cependant l'auteur souligne que la réussite de la méthode SBET dépend fortement de l'expertise et les

connaissances de domaine d'affaires des membres de l'équipe de test. Petty (2005) souligne aussi que la capacité d'apprentissage est plus rapide en TE qu'en TS. Mais selon l'auteur cette capacité dépend encore des personnes impliquées.

L'innovation de la méthode présentée par Petty (2005) est l'utilisation de paires composées des testeurs et des développeurs. Cela permet de corriger les défauts pendant les tests sans avoir le besoin de les reproduire ultérieurement. La participation de testeurs dans la phase de clarification et de définition d'exigences a permis d'améliorer la qualité de l'oracle de test. Toutefois, l'auteur n'a pas présenté de données quantitatives sur la productivité de l'approche SBET et le degré d'amélioration réalisé par l'introduction de l'approche dans la méthodologie de test. En plus, la taille de l'entreprise où s'est déroulée l'expérience est petite ce qui simplifie énormément la communication et la collaboration entre les testeurs et les développeurs. Par contre dans les grandes entreprises, le projet de test est souvent séparé du processus de développement (Pyhajarvi et al., 2003). Par conséquent, nous ne pouvons pas savoir si la façon d'adoption de l'approche SBET proposée par Petty (2005) pourrait s'appliquer dans les grandes entreprises. Cependant, cette étude est un apport utile à l'enrichissement de la littérature sur les modes d'adoption du TE surtout dans les petites entreprises de développement du logiciel.

4.3 Étude de Lyndsay et van Eeden (2003)

Les auteurs Lyndsay et van Eeden (2003) décrivent leur expérience réussie dans la mise en oeuvre de la technique Session Based Exploratory Testing (SBET). Cette mise en oeuvre a pris place dans une petite entreprise anglaise, en s'inspirant des travaux de Jonathan Bach (2000) qui sont déjà abordés dans le chapitre III. L'objectif principal de l'implémentation de l'approche SBET était d'introduire le contrôle et la mesure sur l'activité de test ad-hoc existant dans l'entreprise (test exploratoire libre selon Bach (2003), parce que les testeurs enregistrent les défauts détectés dans le Bug Tracking System). Le choix de la technique SBET était pour répondre à un mandat d'amélioration de la qualité d'une petite application Web déjà testé en utilisant le test ad hoc, tout en restant dans la limite des ressources existantes. Alors le manque du temps et de personnel ont poussé les auteurs à utiliser l'approche SBET au lieu d'utiliser un TS que les ressources disponibles ne le permettent pas.

La méthode proposée par Lyndsay et van Eeden (2003) se fonde sur les quatre éléments clés cités ci-dessous:

Contrôle de la portée du test : Pour gérer la portée de test, les auteurs ont introduit le concept de point de test. Le terme *point de test* sous-entend une partie ou plusieurs concepts du logiciel sous test nécessitant l'exploration et la conception de plusieurs cas de test pour remplir la mission de test de ce point. L'objectif de l'introduction de ce concept est d'avoir le contrôle sur la portée de test pendant le test de chaque version du logiciel. Autrement dit, être capable de déterminer ce qui doit être testé dans chaque version. En effet, l'absence d'une liste de test déterminée dans le processus ad hoc existant et l'absence des exigences dans l'ensemble de projet du développement a rendu difficile l'identification du volume de test nécessaire de chaque version ou après chaque changement. En effet, avant l'introduction de l'approche SBET, la portée de test a été guidée par les défauts trouvés et par les prédictions et les intuitions de testeurs sur les secteurs du logiciel devant être testés. Donc une liste de point de test va permettre de sélectionner les parties devant être testées de chaque version. Ainsi, une liste de points de test va permettre d'évaluer le statut et la progression de projet de test, simplifier la communication à l'intérieur et à l'extérieur de l'équipe de test, d'éviter la duplication du travail à l'intérieur de l'équipe de test dans le sens où une partie pourrait être testée par plusieurs testeurs (Lyndsay et van Eeden, 2003).

Contrôle du travail de l'équipe de test: les auteurs ont proposé le concept de test en session pour contrôler le travail accompli par chaque testeur. La session est un intervalle non interrompu. Dans une session de test, le testeur se charge de l'exécution d'une ou plusieurs points de test et rapporte les défauts trouvés et les questions rencontrées pendant l'exploration à la fin de la session de test. Les questions mènent souvent à des nouvelles pistes pour la conception de d'autres points de test. Ce rapport de test fera l'objet d'une revue et une discussion avec le responsable de test. Ce dernier évalue le travail accompli par le testeur et le guide vers d'autres astuces ou stratégies, si nécessaire (Lyndsay et van Eeden, 2003).

Mesure de la couverture de tests: selon Lyndsay et van Eeden (2003), la couverture de test consiste à mesurer ce qui a été testé comme proportion de ce qui pourrait être testé.

Selon ces mêmes auteurs l'absence des exigences documentées et de cas de test scénarisé a rendu impossible d'utiliser les méthodes formelles habituelles de mesure de couverture de test dans l'activité de test effectué par l'approche de test SBET (ces méthodes seront abordées en détail dans le chapitre VII). Face à ceci, les auteurs ont proposé une technique de mesure de couverture de tests qui s'adapte avec les caractéristiques spéciales de l'approche SBET. Leur technique fondée sur l'estimation et l'évaluation subjective de «la testabilité» sous-entend le pourcentage testé ou couvert par les tests de chaque point de test dans la session de test. Après l'exécution de la session de test, le responsable de test évaluera le travail accompli par le testeur et en même temps vérifiera le pourcentage estimé de la testabilité rapporté par le testeur de chaque point de test. Si le pourcentage est insuffisant et le risque associé à ce point de test exige un pourcentage supérieur, l'effort de test nécessaire pour accomplir ce point de test est re-estimé (Lyndsay et van Eeden, 2003). En calculant la testabilité de chaque point de test, les auteurs ont pu calculer la couverture globale du produit logiciel sous test à n'importe quel moment du processus de test en utilisant la formule suivante:

Couverture de test = la somme de temps de points de test complétés/l'estimation de la somme de temps nécessaire pour accomplir les points de test restants

Mesure et hiérarchisation de risque: les auteurs, Lyndsay et van Eeden (2003) ont mesuré le risque de chaque point de test en terme de la probabilité d'occurrence d'une défaillance associée à ce point de test, et l'impact de cette défaillance sur le fonctionnement du logiciel. Cela leur permis de classifier et d'estimer l'effort nécessaire pour tester chaque point de test, et de prioriser les tâches du test. Autrement dit, les points de test représentant plus de risque recevront plus de tests.

Selon les auteurs Lyndsay et van Eeden (2003) l'introduction de l'approche a eu des résultats immédiats et des résultats à long terme. En ce qui concerne les résultats immédiats, l'équipe de test a pu produire une métrique de couverture utile dès la première utilisation de l'approche SBET. Ce fait a eu une répercussion positive sur la qualité du produit, parce que les parties à grands risques du logiciel ont reçu plus d'attention et plus de tests. L'utilisation

de l'approche SBET a permis de contrôler le travail des testeurs après l'exécution de chaque session sans avoir le besoin d'être sur place pendant le test. En ce qui concerne la productivité, les auteurs n'ont pas pu tirer de conclusions fiables, à cause de l'absence des mesures quantitatives avant l'introduction de l'approche SBET.

Cependant, même avec l'augmentation du nombre d'erreurs trouvées dans les cinq mois qui suivent l'utilisation de SBET, les auteurs Lyndsay et van Eeden (2003) n'ont pas pu savoir si cette augmentation due à l'introduction de l'approche SBET ou l'augmentation de la complexité de l'application et l'ajout de nouvelles fonctionnalités. A long terme, les auteurs Lyndsay et van Eeden (2003) ont remarqué que le produit est devenu plus stable, et que le nombre de défauts trouvés a diminué d'une façon significative bien que de nouvelles fonctionnalités s'ajoutent toujours. Aussi, ils n'ont pas pu savoir si cette réduction provenait de l'amélioration de la qualité du code ou de l'incapacité de l'approche SBET à détecter d'autres défauts. Toutefois, selon Lyndsay et van Eeden (2003) l'introduction de la technique a eu une répercussion positive sur tout le projet de développement, du fait qu'elle a incité les responsables de projet de développement à améliorer la globalité du processus de développement, surtout la documentation. Selon ces mêmes auteurs, quelques résultats intangibles ont été perçus suite à l'introduction de SBET. Il s'agit de l'amélioration de la visibilité à l'intérieur et l'extérieur du processus de test.

En général, selon les auteurs Lyndsay et van Eeden (2003) l'approche SBET a été très efficace dans le cas de leur projet de test. Elle a permis d'introduire le contrôle et la mesure sur le processus ad hoc existant. Ces mêmes auteurs soulignent que la méthode a permis d'encourager la communication entre les membres du test au lieu d'utiliser la documentation pour le faire. Ils ajoutent que le débriefing utilisé dans l'approche SBET après l'exécution de chaque session de test a permis de former les testeurs et leur apprendre les techniques de tests. Toutefois, ils affirment que cette méthode pourrait être moins efficace dans un environnement de développement plus sophistiqué. Ils soulignent aussi que la qualité de test effectué dépend de la créativité et l'expertise des membres de l'équipe de test.

La taille et la nature de l'application qui a été le sujet de cette étude ne permettent pas de généraliser les résultats de cette étude. La métrique de couverture proposée dans l'étude est subjective et dépend aussi de l'expertise du testeur. Mais les idées et les techniques de mesures proposées sont intéressantes et initient plusieurs pistes de recherches, afin d'améliorer l'adoption de l'approche SBET.

4.4 Étude de James et Wood (2003)

Les auteurs Wood et James (2003) décrivent leur expérience d'utilisation de l'approche Session Based Exploratory Testing (SBET). Alors l'approche SBET a été introduite pour tester un logiciel destiné à être utilisé dans les appareils médicaux. Les auteurs ont eu recours à la technique SBET pour deux raisons: la première raison est la moindre coût de l'approche SBET qui leur a permis de l'utiliser comme une méthode complémentaire à la méthode scénarisée de test. Cette dernière a un coût considérable à cause des frais de documentation des tests. Cette documentation doit être détaillée et rigoureuse dans le domaine du test des logiciels médicaux, afin de respecter les normes de la qualité du système (Quality System Regulation). La deuxième raison est que les auteurs ont eu besoin d'une méthode de test pouvant introduire l'innovation et la créativité dans le test, en leur permettant de détecter les erreurs manquées par l'approche scénarisée.

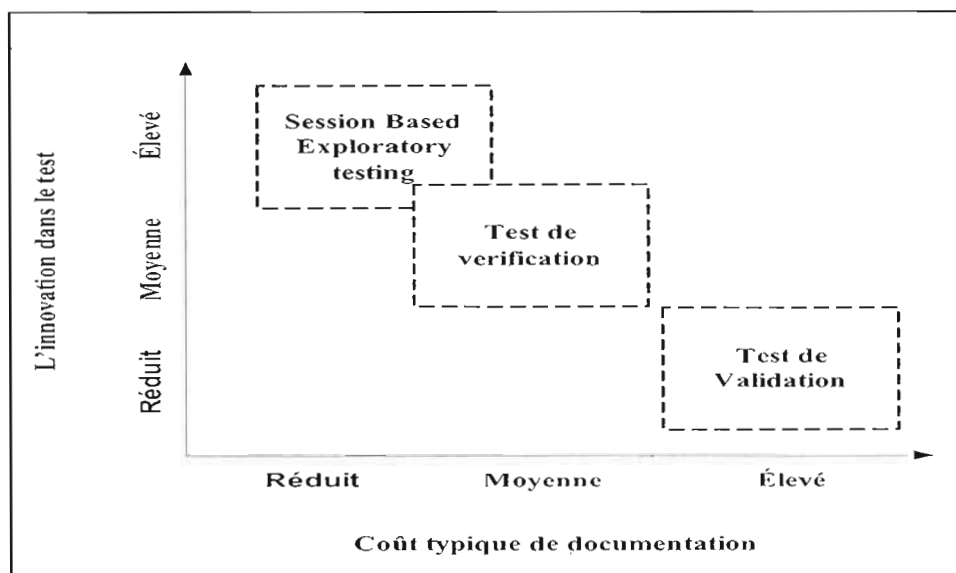
Les auteurs ont utilisé l'approche SBET comme une méthode de test complémentaire à la méthode scénarisée principale. Cette dernière est basée sur la validation des exigences et la vérification de code du logiciel sous test. Le test de validation est centré sur la couverture des exigences et le respect des standards. Ceci nécessite une traçabilité accrue entre les procédures de tests et les exigences initiales. Selon James et Wood (2003) ce type de test ne met pas l'emphasis sur la détection des défauts tant que le respect des exigences et les normes du domaine de développement du logiciel médical. Le test de vérification est basé sur la couverture de code. Ce type de couverture cherche à satisfaire un critère de couverture de code, par exemple 80% des blocs de code doivent avoir au moins un test qui les parcourt. Autrement, l'exécution d'un maximum de lignes de code possibles pour éviter qu'un bogue reste dans le logiciel à cause d'une ligne de code non exécutée pendant les tests. Selon James et Wood (2003) le test de vérification ne met pas l'accent sur la détection de défauts tant que la couverture de code par les tests.

Les faiblesses des deux méthodes de test, le test de validation et le test de vérification, ont motivé les auteurs à introduire l'approche SBET, dans le but de détecter les défauts qui peuvent être manqués par ces méthodes de test.

James et Wood (2003) ont organisé le test en s'inspirant de la méthode proposée par Jonathan Bach (2000) déjà présentée dans le chapitre III. Au début, ils ont planifié les chartes de test et ont estimé l'effort nécessaire pour remplir chacune d'entre elles. Pendant l'exécution de chaque charte, le testeur devrait enregistrer les défauts détectés et les opportunités rencontrées. Notant qu'une opportunité sous-entend des nouvelles pistes de test découvertes pendant l'exploration, qui pourraient donner lieu à la création de nouvelles chartes. À la fin de l'exécution de chaque charte, les auteurs débriefent le testeur, pour évaluer les résultats rapportés et mettre à jour le plan des chartes si nécessaire.

Selon Wood et James (2003) l'approche SBET est une méthode de test très efficace dans le domaine de développement des logiciels médicaux, du fait qu'elle pourrait détecter les défauts pouvant être manqués par les autres méthodes de test. Un autre avantage de la méthode SBET signalé par ces mêmes auteurs est la documentation produite pendant les tests qui est très appréciée dans le domaine de développement du logiciel médical. Les auteurs soulignent que l'approche SBET a encouragé la créativité et l'innovation de testeurs. Cela a permis de détecter des défauts importants dans le logiciel à moindre coût par rapport aux méthodes scénarisées traditionnelles, tel qu'illustré à la figure 4.1.

Figure 4.1 : Coût de documentation versus l'innovation dans le test (Adapté et traduit de James et Wood)



En ce qui concerne la productivité de l'approche SBET, les auteurs James et Wood (2003) soulignent qu'elle est plus efficace que les deux approches scénarisées utilisées, le test de vérification et le test de validation en se basant sur les résultats publiés par (Jones, T.C.Jones, 1998). Ces résultats ont montré que le test d'utilisabilité est plus efficace que le test de vérification et le test de validation. En faisant l'analogie entre le test d'utilisabilité et le SBET, les auteurs ont pu conclure que l'approche SBET est plus efficace que la méthode scénarisée, c'est à dire plus efficace que le test scénarisé de validation et de vérification. En effet, selon James et Wood (2003), le test d'utilisabilité et l'approche SBET sont semblables à cause de trois raisons: la première est que les deux se fondent sur le manuel d'utilisateur, la deuxième que les deux s'effectuent sur des petites périodes séparées, connues sous le terme de «session de test» et la troisième que les deux utilisent les talents et la créativité de testeurs.

Toute fois, James et Wood (2003) ont souligné que la qualité de test effectué dépend des qualifications et l'expertise des testeurs qui exécutent les tests. Selon les auteurs la méthode SBET ne fournit qu'un protocole ou une structure d'organisation et de gestion, mais ne garantit pas la qualité de test effectué. Ils ont souligné aussi que le rôle de responsable de test est crucial et influence la qualité globale de test effectué, du fait que c'est lui qui identifie et

détermine la liste des éléments de test et le contenu des chartes. Par conséquent la couverture de test de haut niveau dépend des compétences et de l'expertise du responsable du test.

Cette étude montre que le TE pourrait être utilisé même dans les domaines de test le plus critique, comme une méthode complémentaire à la méthode scénarisée. À cause de sa valeur ajoutée et son innovation dans la détection des défauts pouvant être manqués par les méthodes scénarisées traditionnelles. Cependant, l'étude n'a pas présenté des données quantitatives sur la productivité de l'approche SBET et le degré d'amélioration réalisé de qualité du logiciel testé.

4.5 Étude de Amland et Vaga (2002)

Les auteurs Amland et Vaga (2002) décrivent une étude de cas d'utilisation de TE en tant qu'une stratégie de test primaire dans une entreprise norvégienne. L'introduction de l'approche était pour tester un portail Web. L'instabilité et le changement fréquent des exigences et le manque du temps, étant les motifs principaux qui ont poussé les auteurs à chercher une approche de test qui peut s'adapter avec les contraintes changeantes de leur projet de test, à la place de l'approche scénarisée habituelle qui était incapable de s'adapter à ces contraintes.

En effet, au début, les auteurs ont commencé la préparation de plan de test et des cas de test formels nécessaires, pour la mise en œuvre d'une approche de test scénarisé. À cause de l'absence des exigences du système, les auteurs ont utilisé le TE libre pour se renseigner sur le logiciel, planifier et concevoir les cas de tests. Cependant, les développeurs ou les auteurs ont eu le mandat de tester le logiciel développé ont continué d'introduire des changements au code. De ce fait, selon Amland et Vaga (2002), l'approche scénarisée de test ne pourra pas être rentable dans leur cas, parce qu'après chaque changement dans le logiciel ils devront retravailler les artefacts de test déjà conçus.

À cet effet, Amland et Vaga (2002) ont décidé d'utiliser le TE. Cependant, les auteurs ont réalisé que la méthode de gestion et de contrôle de TE a un rôle décisif dans la réussite de leur projet de test, surtout si tout le temps disponible pour le test est seulement de deux jours.

Alors, Amland et Vaga (2000) ont géré le TE d'une façon proche de la méthode proposée par Jonathan Bach (2000). Ils ont préparé des directives pour les sept pairs participantes dans le test d'acceptation de portail. En fait, ces directives ont été élaborées à partir de plan de test formel déjà conçu. Ce plan identifie déjà les items du logiciel à tester et les items ne doivent pas être testés. Ces items ont été utilisés pour contrôler la couverture de tests. Avant le début de test les testeurs ont suivi une formation de deux jours, puis un briefing de 20 minutes a été mis en oeuvre pour annoncer aux testeurs les secteurs fonctionnels principaux qu'ils doivent tester. En plus, un contrôle auprès de testeurs pendant le déroulement de test a aidé les auteurs dans la direction des testeurs en cas de déraillement sur les directives.

Selon les constatations de Amland et Vaga (2002) l'expérience d'utilisation de TE a été fructueuse. Toutefois, ils affirment que la créativité et les connaissances des testeurs et du responsable du test chargé de la sélection de la liste des items à tester sont essentielles dans le TE et peuvent influencer la qualité du test.

Malgré la réussite de cette étude de cas, la taille et la nature de l'application ne permettent pas de généraliser les résultats obtenus, ni de tirer des conclusions fiables. Toutefois, cette expérience a introduit une situation réelle où le TE a été implémenté pour confronter les contraintes changeantes de projet de test. Cette étude de cas nous a montré que les artefacts scénarisés formels pourront servir dans l'implémentation de TE sans avoir l'obligation d'utiliser les techniques proposées par Jonathan Bach (2000) et Lyndsay et van Eeden (2003).

4.6 Synthèse des résultats des études proposées

Les études que nous avons présentées dans ce chapitre nous ont permis de tirer plusieurs conclusions d'après des expériences réelles d'utilisation de TE. Ainsi, elles nous ont permis de comprendre comment les praticiens et les professionnels dans l'industrie de test perçoivent le TE, comment ils l'implémentent dans la pratique, pourquoi ils l'utilisent, les difficultés et les lacunes rencontrées lors de l'utilisation de TE, et finalement d'élaborer une vue globale et commune à partir de toutes les études proposées.

Nous avons constaté que les praticiens ne considèrent que l'approche SBET comme un TE tandis que le TE libre est considéré comme un test ad hoc (Lyndsay et van Eeden, 2003; Amland et Vaga, 2002). Ainsi, tous les auteurs dans les études de cas que nous avons présentées adoptent une forme personnalisée de l'approche SBET. Autrement dit, les auteurs organisent l'activité de test dans des sessions de test de durée déterminée, chacune d'elles produit des notes qui font l'objet d'une évaluation par le responsable de test. Nous avons constaté aussi que ces études ne mentionnent pas les techniques utilisées pendant l'exploration et les tests, malgré la diversité des techniques proposées par les concepteurs de TE Kaner et Bach (2004) dont quelques-unes déjà évoquées dans le chapitre III. En fait, dans la plupart des études, les testeurs utilisent leurs intuitions pour détecter les défauts, nous pourrions même dire qu'il s'agit d'un test ad hoc planifié et structuré.

Toutes les études que nous avons présentées dans ce chapitre ont montré que les changements fréquents du logiciel, la pression de temps et la limitation de ressources en terme de budget de test et de personnel sont les raisons principales pour utiliser le TE, plus particulièrement l'approche SBET. Certaines études (Itkonen et Rautiainen, 2005; Lyndsay et van Eeden, 2003; Amland et Vaga 2002) ont signalé le manque de contrôle de couverture de test comme une lacune du TE. Toutes les études (Itkonen et Rautiainen, 2005; Petty, 2005 ; Lyndsay et van Eeden, 2003; Wood et James, 2003; Amland et Vaga, 2002) ont signalé que la qualité du TE effectué dépend des qualifications et de la créativité des testeurs. De plus, deux de ces études ajoutent que la qualité de TE dépend aussi des compétences et des qualifications du responsable de test (Wood et James, 2003; Amland et Vaga, 2002). La planification et le contrôle de l'approche SBET ont été mentionnés comme des facteurs importants de succès de l'activité de test (Lyndsay et van Eeden, 2003; Wood et James, 2003; Amland et Vaga, 2002).

En général, nous avons constaté que toutes les études de cas ont été faites sur des petites applications, et avec des petites équipes de test. La collaboration et la communication entre les membres de l'équipe de test, la concentration sur l'accomplissement du travail de test plutôt que la documentation et la gestion de processus ont été les éléments clés de l'approche SBET. Ceux-ci, nous ont poussé de faire l'analogie entre le développement du logiciel et le

test du logiciel, autrement dit, l'analogie entre l'agilité et la discipline du processus de développement et l'informalité et la discipline de processus de test. Nous allons aborder en détail au cours de notre étude comparative dans le chapitre VII cette analogie que nous avons l'exploitée pour construire notre cadre conceptuel de comparaison qui va nous permettre de comparer les deux approches de test : le TE et le TS.

CHAPITRE V

L'ÉTUDE EMPIRIQUE

Dans ce chapitre nous présentons les étapes principales de notre étude empirique. Tout d'abord nous proposerons la motivation de l'étude et la stratégie que nous avons choisie pour l'aborder. Puis nous présenterons le schéma conceptuel de l'expérience. Par la suite nous analyserons les résultats recueillis et nous conclurons.

5.1 Motivation de l'étude

Depuis son apparition dans l'industrie du test, le test exploratoire (TE) se fait présenter comme une approche productive qui pourrait augmenter l'efficacité de l'activité de test en termes de nombre et d'importance de défauts détectés. Selon Bach (2003) le TE pourrait être plus productif que le test scénarisé (TS). Cependant, l'auteur n'a pas présenté de preuves de cette réclamation, à part quelques anecdotes et expériences personnelles. Un tour rapide sur les récentes publications de Kaner sur son site¹, montre que le TE se fait traiter comme une innovation scientifique qui exploite et optimise la créativité et l'expertise du testeur dans la détection des défauts importants qui ne pourraient pas être détectés par le TS. Selon Kaner et Bach (2005), le TS transforme les testeurs en robots inefficaces. Ces arguments nous ont motivée à faire une étude empirique pour évaluer la productivité du TE. Tout d'abord nous allons comparer les résultats de TE recueillis de l'expérience avec les résultats quantitatifs publiés par Itokens et Rautiainen (2005). Puis nous procéderons à une analyse comparative empirique entre le TE et le TS en se basant sur les résultats de notre étude.

Cependant, dans la mise en œuvre de notre étude empirique nous avons été confrontée au

¹ www.testingeducation.org

problème du contexte expérimental. En effet dans un contexte industriel, nous pouvons utiliser comme instrument de l'expérience un logiciel professionnel, c'est à dire un logiciel développé dans l'industrie de développement du logiciel. Ce logiciel pourrait être testé avec deux groupes, un utilise le TS et l'autre le TE. Les données pourraient être recueillies pendant l'activité de test et sur le site de production du logiciel testé. Par la suite, l'analyse des résultats de l'expérience doit être faite sur deux niveaux : le premier est de comparer le nombre et l'importance des défauts détectés avant la livraison du logiciel, c'est à dire à la fin de l'activité de test. Le deuxième, est de comparer le nombre et l'importance des défauts détectés après la livraison du logiciel, c'est à dire dans le site d'utilisation du logiciel testé. Cette stratégie permettrait de mesurer la productivité pendant et après l'activité de test. Or, la mise en place d'une telle expérience nécessite l'engagement d'une entreprise de développement du logiciel à participer à l'expérience et à divulguer les informations de son activité de test. Malheureusement, nous n'avons pas pu trouver une entreprise pour se plier à ces contraintes. Cela nous a obligée à concevoir une nouvelle stratégie pour notre expérience dans le contexte académique où nous avons décidé de la faire.

5.2 La stratégie de l'expérience

Dans un contexte académique, l'expérience pourrait être entreprise de deux façons différentes : la première consiste à faire l'expérience de la même façon que si elle se déroulait dans le contexte industriel, c'est à dire nous divisons les sujets en deux groupes, dont un exécute le TE, et l'autre le TS. Nous pouvons même aller plus loin et utiliser l'approche SBET pour contrôler la couverture de test et éviter que les défauts détectés soient dupliqués. Quant au TS, il pourrait être exécuté de la même façon qu'en industrie. Alors, chaque sujet exécute des cas de tests correspondant à une partie du logiciel. Finalement nous analysons le nombre et l'importance des défauts rapportés pour déterminer l'approche la plus efficace.

Malheureusement, nous n'avons pas pu utiliser cette stratégie pour deux raisons: la taille du programme utilisé dans l'expérience et le manque d'expérience des expérimentateurs. En effet, nous avons dû utiliser un petit programme dans l'expérience afin de simplifier la mission des sujets pendant l'activité de TE. Cela pour éviter d'obtenir des résultats nuls qui peuvent résulter de l'exécution de TE si nous utilisons un trop grand logiciel.

La deuxième raison du choix de notre stratégie est le manque d'expérience chez les participants. Ces sujets sont des étudiants à l'UQÀM. Ils possèdent une expérience des tests et de ses techniques, limitée à la couverture de ces matières dans le programme offert à l'UQÀM. Nous avons cependant choisi les étudiants du cours INF6160 *Qualité : processus et produits*, parce que c'est dans ce cours que ces sujets sont abordés le plus profondément. Cela ne nous a quand même pas permis d'organiser l'activité de test de la même façon professionnelle décrite ci-dessus. L'expérience consiste à utiliser les mêmes sujets pour exécuter d'abord le TE et le TS ensuite, afin d'éviter que les sujets apprennent les cas de tests scénarisés et les répètent par la suite dans le TE. À cet effet, nous avons programmé l'expérience dans une séance de cours de 2 heures, 45 minutes pour exécuter chacune de deux approches de test. Les étudiants ont pris connaissance du déroulement de l'expérience dans une séance antérieure, où le professeur a présenté aux étudiants une vue d'ensemble du TE. Le sujet de l'expérience et ses résultats a constitué une partie de travail de session de cours INF6160. Alors, chaque étudiant participant dans l'expérience a dû rapporter ses résultats et les conclusions qu'il a pu tirer de l'expérience concernant les deux approches de test : le TE et le TS dans un rapport de fin de session. Ceci a motivé les sujets à détecter le maximum des défauts possibles et nous a garanti que les étudiants ont pris l'expérience au sérieux.

Nous avons proposé aux 56 sujets participants dans l'expérience le programme à tester accompagné de quelques modèles et techniques d'exploration (Annexe C), pour les guider pendant le TE et éviter d'obtenir des résultats nuls. Puis nous leur avons proposé les cas de tests scénarisés que nous avons préparé pour l'activité de TS. Alors, tous les sujets ont exécuté les mêmes cas de tests.

Dans notre plan expérimental, les mêmes sujets ont été utilisés dans les deux traitements. Ce type d'étude est qualifié de plan expérimental à facteur unique à mesures répétées sur les mêmes éléments « Single Factor Experiments Having Repeated Measures on The Same Elements » (Winer, 1971, p. 105). Les résultats ont été exploités de deux façons différentes : la première, l'exploitation des résultats de TE collectés de notre expérience, et la deuxième l'exploitation des résultats des deux activités de test et la comparaison des résultats collectés. À cet effet, nous utilisons dans cette analyse comparative l'analyse de variance **ANOVA**

proposé par Winer (1971, p. 105). Cette analyse nous a permis d'évaluer l'effet de traitement, c'est à dire l'approche de test (TS, TE) sur la performance des sujets. Cette façon de faire nous a permis d'évaluer la productivité de chacune des deux approches de test. Cela a aussi permis d'explorer la validité de l'hypothèse proposée par Kaner et Bach (2005) qui cite que les testeurs juniors ne pourraient pas reproduire les cas de tests de la même façon que l'auteur ou le concepteur de ces cas de tests (le testeur senior). Aussi, nous a permis d'analyser et de comparer les résultats de TE de notre étude empirique avec les résultats des travaux de recherches proposés dans la littérature par Itokens et Rautiainen (2005).

5.3 Le schéma de l'expérience

5.3.1 Objectifs et hypothèses de l'expérience

Comme le soulignent Lott et Rambach (1997) l'identification précise des buts de l'étude est essentielle à la mise en œuvre d'une étude expérimentale. Cette identification permet de planifier et déterminer les détails de la perspective ou la stratégie suivie pour que l'expérience atteigne ses buts spécifiés. De ce fait, les aspects à évaluer et les métriques à utiliser devront être précisés et bien identifiés avant le début de l'expérience. Dans notre cas, le but de l'expérience est de comparer la productivité de TE et le TS en termes de nombre et d'importance des défauts détectés. Cela nous a permis d'énoncer les hypothèses suivantes :

Notre hypothèse primaire est :

H1 – le test exploratoire est plus efficace, en terme de nombre de défauts détectés, que le test scénarisé

L'hypothèse nulle correspondante à cette hypothèse est :

H0 – il n'y a pas de différence entre le test exploratoire et le test scénarisé quant au nombre de défauts détectés

Notre hypothèse secondaire est :

H2 – le test exploratoire permet de détecter des défauts plus importants, point de vue gravité, que le test scénarisé

5.3.2 La conception expérimentale

Dans cette section nous présentons la conception expérimentale que nous avons faite pour notre étude empirique. La conception et l'analyse de l'expérience sont traitées complètement par (Winer, 1971) qui été utilisé comme référence dans plusieurs études expérimentales antérieures (Wood et al., 1997). Avant d'introduire les détails de la conception choisie, nous définissons certains termes nécessaires à la compréhension de la conception de notre expérience.

- **Variable indépendante:** est le facteur qui influence les résultats de l'expérience, c'est à dire le facteur causal. La manipulation des valeurs prises par cette variable permet d'étudier les effets de ces différentes valeurs sur les résultats. Dans notre étude la variable indépendante est l'approche de test et ses valeurs possibles sont le TE et le TS.
- **Variable dépendante:** mesure les effets de la manipulation des variables indépendantes. Dans notre expérience elle correspond au nombre et la sévérité des défauts détectés.

Dans notre expérience nous allons étudier l'effet de l'approche de test (TE, TS) sur la productivité de l'activité de test en utilisant un échantillon composé de 56 sujets. Chaque élément de l'échantillon applique les deux techniques de test sur le même programme à tester. Donc nous avons un seul facteur qui peut influencer les résultats de l'expérience qui est l'approche de test (TE, TS). Selon (Winer, 1971) les contraintes de notre étude empirique correspondent à la conception proposée par l'auteur sous l'expression : « expérience à facteur unique a mesures répétées sur les mêmes éléments » (Single Factor Experiments Having Repeated Measures on the Same Elements).

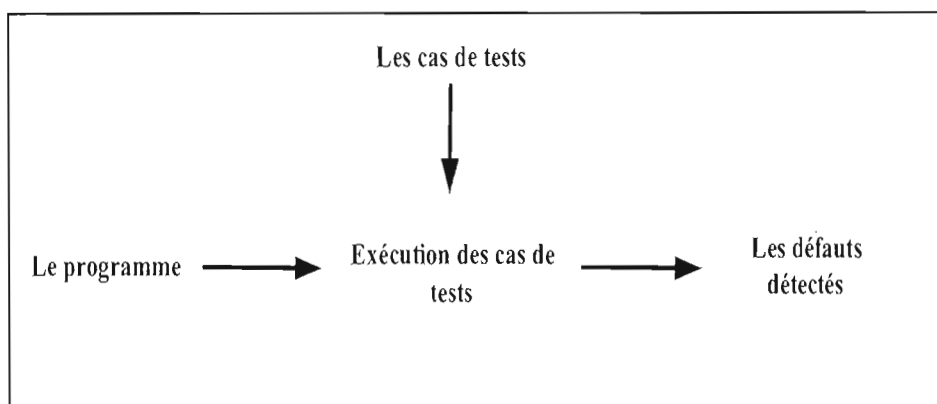
5.3.3 Les instruments de l'expérience

Les instruments de l'expérience sont constitués d'un seul programme dans les deux activités de test le TE et le TS. Il s'agit d'un programme de gestion des messages développé avec le langage de programmation Java. Nous avons choisi le programme pour que sa taille et sa complexité facilitent l'exécution des deux techniques de test aux sujets (les étudiants de cours INF 6160).

5.3.4 Le traitement expérimental

En ce qui concerne le TS, les sujets ont reçu en entrée, le programme et les cas de tests que nous leur avons conçus en utilisant un test fonctionnel (boîte noire). Les sorties sont les défauts détectés.

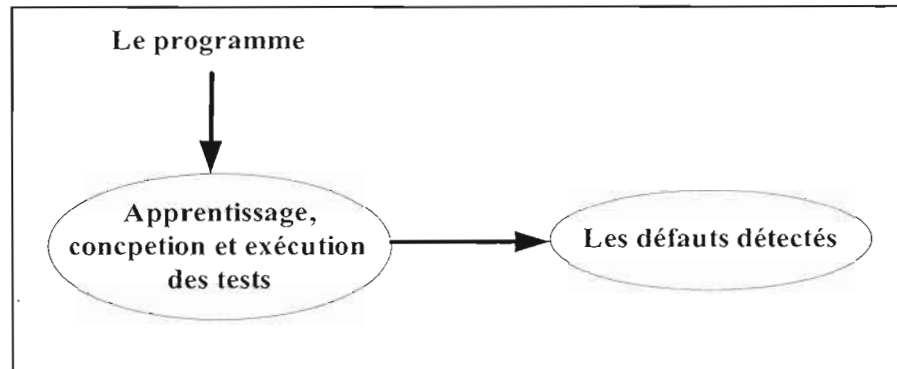
Figure 5.1 : Le traitement de test scénarisé



Tel qu'illustré sur la figure 5.1, les sujets ont reçu les cas de test et les ont exécutés dans une session de 45 minutes. Ils ont été informés de ne pas produire de cas de tests additionnels pendant l'exécution des cas de tests, pour éviter d'introduire le TE dans le TS. Alors pendant l'exécution des cas de test, les sujets comparent les résultats de sorties observés avec les résultats décrits dans le script de cas de test. Si les deux résultats ne correspondent pas, le sujet doit enregistrer et décrire le comportement observé pendant l'exécution de ce cas de test, pour que nous puissions nous assurer qu'il s'agit vraiment d'un défaut et éviter une mauvaise interprétation du comportement du programme.

En ce qui concerne le TE, nous l'avons programmé dans une session de 45 minutes. Les sujets dans cette session de test exploratoire ont utilisé quelques modèles et techniques d'exploration que nous leur avons préparés en avance (Annexe C), en se basant sur les techniques proposées par Amland (2002), afin de faciliter leur mission et les guider dans le test.

Figure 5.2 : Le traitement de test exploratoire



Tel qu'illustré sur la figure 5.2 les sujets ont reçu le programme en entrée. Leur mission était d'apprendre le programme, concevoir et exécuter les tests et rapporter les défauts détectés. Le résultat de l'exécution de TE est une liste des défauts détectés (Annexe D). Pendant, l'exécution de TE les sujets ont classifié les défauts détectés selon leur gravité en suivant une liste de classification de défauts que nous leur avons fournie avant le début de l'activité de TE. Cette liste classifie la gravité des défaillances en trois niveaux:

- ❑ Fatale: l'application est inopérable complètement (Crash)
- ❑ Moyenne: l'application fonctionne, mais certaines fonctions sont inopérables ou certains résultats sont erronés
- ❑ Mineure: l'impact est mineur sur l'utilisation du système comme certains formats sont erronés, bien que les résultats soient corrects, ou encore le délai de réponse est supérieur de ce qui attendu d'une telle application.

Après l'expérience nous avons re-classifié les défauts rapportés par les expérimentateurs pour éviter des erreurs qui peuvent se résulter sur une mauvaise classification.

5.4 Analyse des résultats de l'expérience

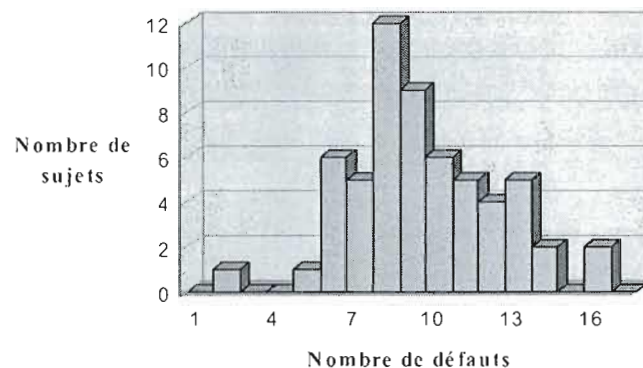
5.4.1 Analyse des résultats de test exploratoire

Avant de procéder à une analyse comparative des résultats de TE et TS nous analysons tout d'abord des résultats de test exploratoire en termes de nombre et l'importance des défauts détectés et nous les avons comparés avec les résultats rapportés dans la recherche d'Itokens et Rautiainen (2005).

5.4.1.1 La productivité de TE selon le nombre de défauts détectés

L'analyse et le traitement des données de TE recueillis pendant l'étude empirique, nous ont permis de développer une idée estimative de la productivité de TE en terme de nombre de défauts détectés (figure 5.3)

Figure 5.3 : Les résultats de test exploratoire

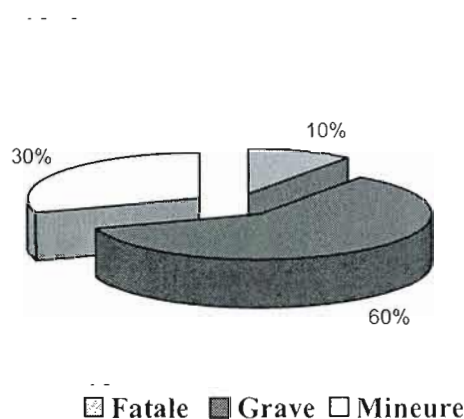


Les résultats présentés à la figure 5.3, montrent que plus que la moitié (66%) des sujets ont réussi à détecter entre 6 à 9 défauts. La moyenne est de 9.5 défauts par sujet. Cette moyenne est très proche de celle proposée par l'étude de Itokens et Rautiainen (2005). Selon les données quantitatives collectées par ces auteurs dans la petite entreprise, dont son contexte de développement est immature (plus proche de notre contexte de test), la moyenne réalisée est de 8.7 défauts dans une session de 60 minutes.

5.4.1.2 La productivité selon l'importance de défauts

L'analyse et le traitement des données de TE recueillis pendant l'étude empirique, nous ont permis d'avoir aussi une idée sur l'importance de défauts qui pouvant être détectés (figure 6.4).

Figure 5.4 : Importance de défauts détectés avec le test exploratoire



Nous pouvons constater à partir des résultats présentés sur la figure 5.4 que le pourcentage des défauts graves détecté par les sujets avec le TE est plus que la moitié de tous les défauts détectés. Tandis que seulement 10% des défauts détectés sont fatales. Les auteurs Itokens et Rautiainen (2005) ont rapporté un pourcentage de 15% des défauts fatals détectés dans une session de 60 minutes. C'est un pourcentage très proche du pourcentage réalisé dans notre étude empirique, si nous prenons en considération la différence dans les programmes utilisés dans leur étude de cas et notre expérience.

La figure 5.4 montre que 70% des défauts détectés pendant l'expérience avec le TE sont des défauts importants, c'est à dire sont des défauts fatals ou graves, qui pourront empêcher le fonctionnement normal du programme sous test. Par contre, 30 % des défauts détectés avec le TS sont des défauts mineurs c'est à dire des défauts qui ne pourront pas empêcher le programme de fonctionner. Ainsi, la moitié des défauts détectés avec le TS sont des défauts

importants, dont 45% des défauts sont des défauts graves et seulement 5% des défauts sont fatals. De ce fait, nous pouvons conclure que le TE permet de détecter des défauts plus importants que le TS. Évidemment, les résultats de TS dépendent des connaissances et des compétences du concepteur des cas de test (l'auteur de ce travail de recherche). À cet effet, un autre concepteur peut aboutir à des résultats différents.

Pendant l'expérience, nous avons constaté que la boucle d'apprentissage et les feedbacks instantanés durant l'activité de test constituent les raisons principales des résultats performants du TE. En effet, les sujets ont commencé l'apprentissage et la conception des cas de test dès qu'ils ont reçu le programme à tester. Au fur et à mesure qu'ils avancent dans leur mission de TE, ils conçoivent des tests plus productifs et plus performants, qui leur permettent de détecter des défauts plus importants. Cela, grâce aux feedbacks dérivés des tests exécutés ultérieurement pendant l'activité de TE et les connaissances accumulées depuis le début de la mission de TE.

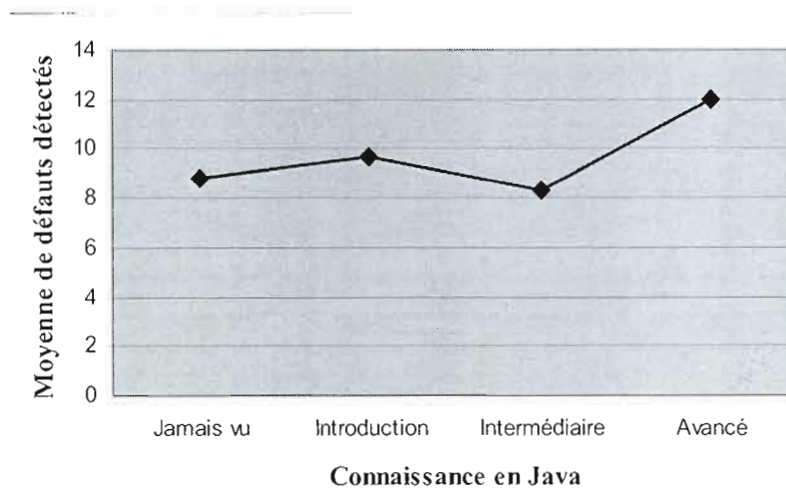
Nous croyons aussi que la diversité des compétences et les qualifications des sujets participants dans l'expérience a contribué aussi aux résultats performants en TE. En fait, la participation de plusieurs compétences ou personnes dans le test donne des résultats meilleurs que lorsque la conception des cas de test est faite par une ou deux personnes seulement. Nous pouvons conclure de cette discussion que le TE permet de détecter des défauts plus importants, point de vue sévérité, que le TS. Autrement, que notre hypothèse secondaire H2 ne peut pas être rejetée.

Cependant, nous avons constaté que quelques sujets dans l'expérience ont pu détecter des défauts plus importants que ceux détectés avec le TS. Nous croyons que l'importance des défauts trouvés avec le TE dépend de la créativité et les qualifications de testeur. À cet effet nous avons étudié dans notre étude empirique l'influence des connaissances en programmation en Java comme un facteur représentant l'expertise et les qualifications de sujet sur les résultats de TE effectué.

En effet, nous avons choisi ce facteur parce que nous croyons que le niveau de connaissance

en java représente bien la familiarité de l'étudiant avec la programmation et les techniques de débogage, donc implicitement son expertise et ses qualifications nécessaires pour exécuter sa mission pendant le TE (figure5.5)

Figure 5.5 : L'influence de l'expertise sur le test exploratoire

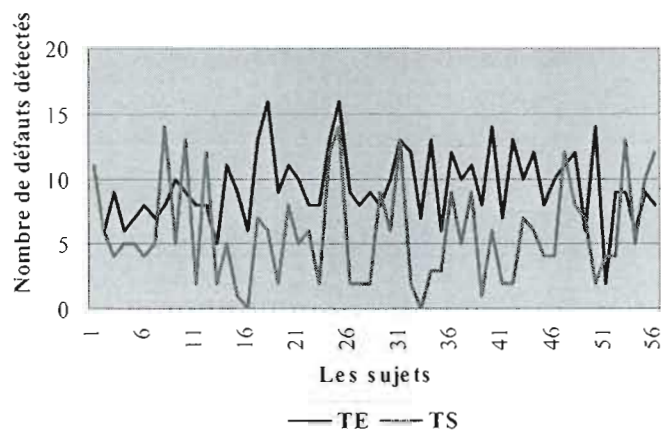


Les résultats présentés à la figure 5.5 montrent que la moyenne de défauts détectés est plus élevée pour un niveau de connaissance élevé en Java. La faible diminution pour le niveau intermédiaire pourrait être expliquée par la difficulté de situer le niveau de connaissance en Java. Notons que la liberté de TE a été signalée par plusieurs sujets comme un avantage du TE qui leur permet d'approfondir et d'améliorer leurs tests en temps réel.

5.4.2 Analyse des résultats de TE et TS

Cette section aborde les résultats rapportés de l'expérience de deux approches de test : le TS et le TE. Notre idée est d'analyser et de comparer la performance des sujets dans les deux méthodes de test. Autrement dit, évaluer la productivité de TE par rapport au TS en comparant le nombre de défauts détectés par chaque sujet en utilisant le TE et le TS. Le traitement des résultats recueillis de l'expérience nous a donné le graphe 5.6

Figure 5.6 : Résultats des sujets aux TE et TS



Nous pouvons constater de la figure 5.6 que le TE est plus productif que le TS. Cependant, la limitation de contexte de l'expérience résidant dans la taille du programme de l'expérience et le manque d'expertise des expérimentateurs ne permet pas de tirer des conclusions fiables et de généraliser les résultats obtenus.

5.4.2.1 Analyse de variance ANOVA

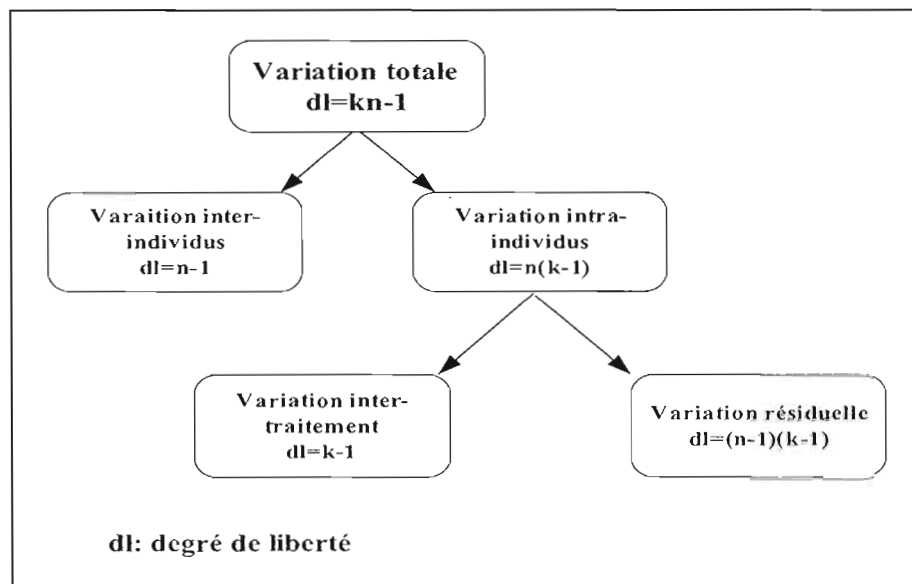
La figure 5.6 montre que les sujets ont été plus performants et productifs en TE qu'en TS. Dans cette section nous allons prouver statiquement ces résultats, en utilisant l'analyse de variance ANOVA.

L'analyse de variance ANOVA « **AN**alysis **Of** **V**ariance» est une méthode statistique permettant de comparer la moyenne de plusieurs populations. Elle permet de savoir si une ou plusieurs variables **dépendantes** sont en relation avec une ou plusieurs variables dites **indépendantes** (Winer, 1971)

Dans un plan d'expérience à mesures répétées, un même sujet est mesuré sous chacun des niveaux d'un traitement expérimental. Comme dans notre étude, où chaque sujet est mesuré deux fois, sous le TE puis le TS. Dans ce type de plan l'effet de l'approche de test

exploratoire sur le sujet k est comparé avec la réponse de même sujet dans le TS. Dans ce plan chaque sujet devient son propre contrôle à travers les deux traitements expérimentaux (les deux approches de test dans notre cas).

Figure 5.7 : Représentation schématique de l'analyse ANOVA (Adapté et traduit de Winer, 1971)



La décomposition de la variance selon (Winer, 1971)

- Variation totale = Variation inter-individus + Variation intra-individus
- Somme carrés totale de la déviation (SC Totale) = Somme carrés inter-individus + Somme carrés intra-individus
- Variation intra-individus = Variation inter-traitement + Variation résiduelle, c'est à dire:

Somme carrés intra individus = Somme carrés inter-traitements + Somme résiduelle des carrés

Dans notre cas nous résumons les calculs de l'analyse de variance ANOVA à mesures

répétées sur les données collectées de notre étude empirique dans le tableau ci dessous :

Tableau 5.1 : ANOVA des données collectées de l'expérience

La source de variation	Somme des carrées SC	dl	Carré moyen	Test-F
Inter-individus	841.45	55		
Intra-individus	837	56		
Inter-traitement	372.96	1	372.96	45.8
Résiduelle	464.04	55	8.14	

La valeur critique pour un **Test F** avec (1.57) degrés de liberté est 7.12. Or, d'après le calcul nous avons trouvé que le **test F = 45.8**. Puisque cette valeur est supérieur de 7.12, nous rejetons l'hypothèse nulle H_0 et nous conservons l'hypothèse H_1 . Or, d'après la représentation graphique des résultats de l'expérience figure 6.6 nous pouvons constater que le TE est plus productif en terme de nombre des défauts détectés que le TS. Par la suite, nous concluons que l'hypothèse H_1 est valide et que le TE est plus productif que le TS.

5.5 Conclusion

L'analyse statistique des résultats de l'étude empirique nous a permis de conclure que la performance des sujets dans l'exécution de TE est supérieure que leur performance dans l'exécution de TS. Par la suite, nous avons conclu que le TE est plus productif en terme de nombre des défauts détectés que le TS. Ainsi, nous avons conclu que le TE pourraient détecter des défauts plus importants, point de vue gravité, que le test scénarisé

CHAPITRE VI

CADRE CONCEPTUEL DE COMPARAISON

Dans la revue de littérature, nous avons compilé quelques études de cas et expériences d'utilisations du TE dans l'industrie de test logiciel. Nous avons présenté un aperçu des raisons que ont motivé les praticiens à utiliser le TE, les façons dont ils ont adopté et géré le TE et les facteurs qui ont influencé ses résultats dans la pratique. Cela nous emmène dans ce travail de recherche à étudier plus profondément et d'une façon générale ces facteurs dans le cadre d'une étude comparative des deux approches de test, le TE et le TS. Dans ce chapitre nous présenterons le contexte de notre étude comparative. Puis nous proposerons notre cadre conceptuel de comparaison. Ce cadre va guider notre analyse comparative de TE et TS. Nous le développerons en se basant sur la littérature et les études de cas des praticiens et chercheurs dans l'industrie de test. À cet effet, les recherches théoriques et empiriques antérieures présentées dans la revue de la littérature seront considérées.

6.1 Mise en contexte de l'étude comparative

Avant de présenter le cadre comparatif de cette analyse, nous proposons le contexte général de notre analyse comparative et les choix que nous avons dû faire, pour rendre possible une comparaison et une discussion cohérente. Tout d'abord, nous choisissons le type de test qui va représenter chacune des deux approches dans cette étude comparative. En effet, étant donné que les deux approches peuvent être représentées sur un continuum (Figure 2.1), la différenciation entre le TE et le TS est difficile et imperceptible sans la spécification de type choisi de chacun d'entre eux. De ce fait nous avons choisi de représenter le TS par un processus de test balisé dans les patrons de standard de documentation IEEE 829 que nous avons proposé dans le chapitre II. Ce choix est motivé par notre besoin de normaliser l'analyse et la comparaison. Ainsi nous pourrons comparer un processus fortement planifié et

discipliné de test avec un autre processus semi-planifié et libre (SBET). Quant au TE, nous avons choisi de le représenter par l'approche Session Based Exploratory Testing (SBET). Cette forme de TE d'après la revue de la littérature que nous avons présenté dans le chapitre IV est la plus adoptée dans l'industrie de test du logiciel comme une méthode primaire de test (Itkonen et Rautiainen ,2005; Petty, 2005 ; Lyndsay et van Eeden ,2003 ; Wood et James, 2003), tandis que le TE libre est toujours considéré comme un test ad hoc (Lyndsay, van Eeden, 2003). Il faut noter que même avec le choix de l'approche SBET nous restons dans la limite de notre comparaison de TE et TS, puisque le TE libre constitue le cœur de l'approche SBET. La seule différence entre les deux types est les notes produites à la fin de la session dans le SBET. Or, ce sont ces notes qui ont favorisé son utilisation dans l'industrie, même là où les tests sont le plus documentés et formels, tel le domaine du logiciel médical (James et Wood, 2003).

L'analyse comparative des deux approches, le TE et le TS, va nous permettre de ressortir les contextes favorables pour utiliser le TE à la place de la méthode scénarisée habituelle de test (TS). En fait, on peut dire qu'un contexte est l'ensemble des facteurs spécifiques de projet de test du logiciel. Par exemple dire que le logiciel à tester est petit représente un facteur de contexte déterminé selon le facteur « Caractéristiques du logiciel à tester ».

Notre comparaison est restreinte au type de test boîte noire, du fait que le TE est une technique de test boîte noire (Craig et Jaskiel, 2002 ; Kaner et al, 2002). Dans ce type de test, on ne s'intéresse pas à l'aspect implémentation du code mais uniquement au comportement du logiciel.

Nous voulons signaler aussi que dans notre analyse comparative, nous ne considérons que les modèles traditionnels de développement. On a vu les modèles agiles de développement constituent un cas réel où le TE et le TS automatisé, sont utilisés ensemble pour tester le logiciel (section 2.4). Or, dans notre étude nous voulons identifier les contextes où le TE pourrait être utilisé comme une méthode primaire de test à la place de TS. À cet effet les modèles agiles ne seront pas considérés dans cette étude.

Dans ce chapitre et les chapitres suivants nous utilisons l'abréviation « TE » pour désigner l'approche SBET, afin d'alléger le texte.

6.2 Cadre conceptuel de comparaison

La comparaison entre le TS et le TE est peu abordée dans la littérature. Les travaux qui ont traité le sujet étaient plus axés sur la proposition et la promotion des avantages de TE par rapport au TS (Kaner, 2006; Bach, 2003). Pour cela, nous nous sommes fixé comme objectif dans ce travail de recherche de comparer d'une façon neutre les deux approches, en mettant l'accent sur les apports et les contextes où le TE pourrait remplacer le TS. Nous tentons par le biais de cette étude comparative d'explorer ces contextes en se basant sur nos déductions personnelles tirés de la base théorique de test du logiciel et la revue de littérature des travaux de praticiens dans l'industrie de test (Chapitre IV), et sur l'étude empirique présentée au chapitre précédent.

Tout d'abord afin que notre analyse comparative soit menée à bien, nous présenterons notre cadre conceptuel de comparaison. Ce cadre doit permettre de guider et focaliser notre analyse comparative entre le TE et le TS. Il regroupe les critères autour desquels la discussion et l'analyse seront centrées. L'identification des facteurs de comparaison a été développée d'une part en se basant sur les résultats des expériences d'utilisation du TE par les chercheurs et les praticiens présentés dans la littérature. Nous nous sommes aussi inspirés des facteurs proposés par Boehm et Turner (2004).

Le cadre proposé par Boehm et Turner (2004) a pour objectif la comparaison des approches agiles et les approches disciplinées de développement du logiciel. Or, notre comparaison pourrait être vue aussi comme la comparaison entre deux méthodes de test, discipliné et agile où le TS représente la méthode disciplinée et le TE la méthode « agile ». On entend par agile le fait que le TE se caractérise par les deux caractéristiques essentielles de l'agilité identifiée dans (Boehm, Turner, 2004). La première étant sa capacité de répondre ou de s'adapter rapidement aux changements du logiciel à tester (Itkonen et Rautiainen, 2005; Petty, 2005; Lyndsay et van Eeden, 2003; Amland et Vaga, 2002). La deuxième est la légèreté de la documentation utilisée et produite pendant le TE.

Le cadre proposé par Boehm et Turner (2004) est axé sur quatre dimensions à savoir. Caractéristiques d'utilisations, caractéristiques de gestion, caractéristiques techniques et caractéristiques du personnel. Or, une méthode de test du logiciel doit être adaptée à un contexte particulier (Craig et Jaskiel, 2002; Perry, 2000) Ce contexte se résulte de l'interaction de plusieurs facteurs reliés aux objectifs principaux de l'activité de test, les ressources financières, techniques, humaines et organisationnelles existantes. À cet effet, le cadre de comparaison de deux méthodes de test se composait de mêmes dimensions citées par Boehm et Turner (2004). Notre cadre de comparaison va regrouper les axes de comparaisons suivantes: caractéristiques d'utilisations, caractéristiques de gestion, caractéristiques techniques et caractéristiques de personnels. Cependant, il nous revient de déterminer les facteurs de comparaison associés à chaque dimension. De plus, nous ajoutons une cinquième dimension traitant la productivité de l'activité de test, puisque la vérification de la productivité de TE est l'un des objectifs principaux de ce travail de recherche. Alors notre cadre conceptuel de comparaison se constitue des dimensions et facteurs suivants présentés dans les sections ci-dessous.

6.2.1 Les caractéristiques d'utilisation

Selon Turner et Boehm (2004) les caractéristiques d'utilisations représentent les aspects et les types de projets appropriés pour chaque approche du développement. Ils ont identifié sous cette dimension les facteurs suivants: les objectifs principaux d'utilisation de chaque approche du développement, la taille de projet du développement en termes de nombre de personnes participants dans le projet, la complexité, le volume du logiciel et le type d'environnement d'affaire où se développe le projet. Dans notre cas, les caractéristiques d'utilisation regroupent les facteurs suivants:

- Les raisons d'utilisation ou les motivations pour utiliser chacune de deux approches de test, le TE et le TS ;
- Les caractéristiques du logiciel à tester en termes de la taille, de la complexité et de la criticité ;
- Le type d'environnement d'affaire où se produit le projet de test ;

- Les ressources financières ;
- Le temps disponible pour remplir l'activité de test.

6.2.2 Les caractéristiques de gestion

Selon les auteurs Boehm et Turner (2004) les caractéristiques de gestion décrivent les façons de gérer le projet du développement dans les deux méthodes du développement, agiles et disciplinées. Ils ont identifié sous cette dimension les facteurs suivants: La planification et le contrôle de projet de développement, la relation avec le client et la communication dans le projet du développement. Dans notre cas de recherche cette dimension de comparaison regroupe les mêmes facteurs discutés par Boehm et Turner mais dans le cadre de projet de test. Il s'agit de la planification de test, le contrôle et le suivi de la progression de test, la communication dans le projet de test et la relation avec le client.

6.2.3 Les caractéristiques techniques

Selon Boehm et Turner (2004) les caractéristiques techniques décrivent comment chacune de deux méthodes du développement, agile et discipliné abordent les étapes essentielles du cycle de développement du logiciel: la spécification des exigences, le développement et le test. Dans notre cas, cette dimension décrit comment les deux approches de test, le TE et le TS abordent les étapes essentielles de l'activité de test. Selon (Pyhajarvi et al, 2003; Swebok, 2004; Craig et Jaskiel, 2002; Perry, 2000) ces activités sont : la planification de tests, la conception de tests, l'exécution de tests. Toutefois, les activités de test découlent, selon (Bolton et al., 2005), de trois facteurs: l'oracle de test, les risques du logiciel à tester et la couverture du test. Ces éléments seront donc discutés sous cette rubrique.

6.2.4 Les caractéristiques du personnel

Les auteurs Boehm et Turner (2004) abordent sous cette dimension les caractéristiques du personnel impliqué dans les projets de développement, qui pouvaient favoriser l'utilisation de chacune de deux approches de développement, agile et discipliné. Ils ont identifié les facteurs suivants: les caractéristiques du client, les caractéristiques des développeurs et la culture de l'organisation où se déroule le projet de développement. Dans notre analyse comparative de

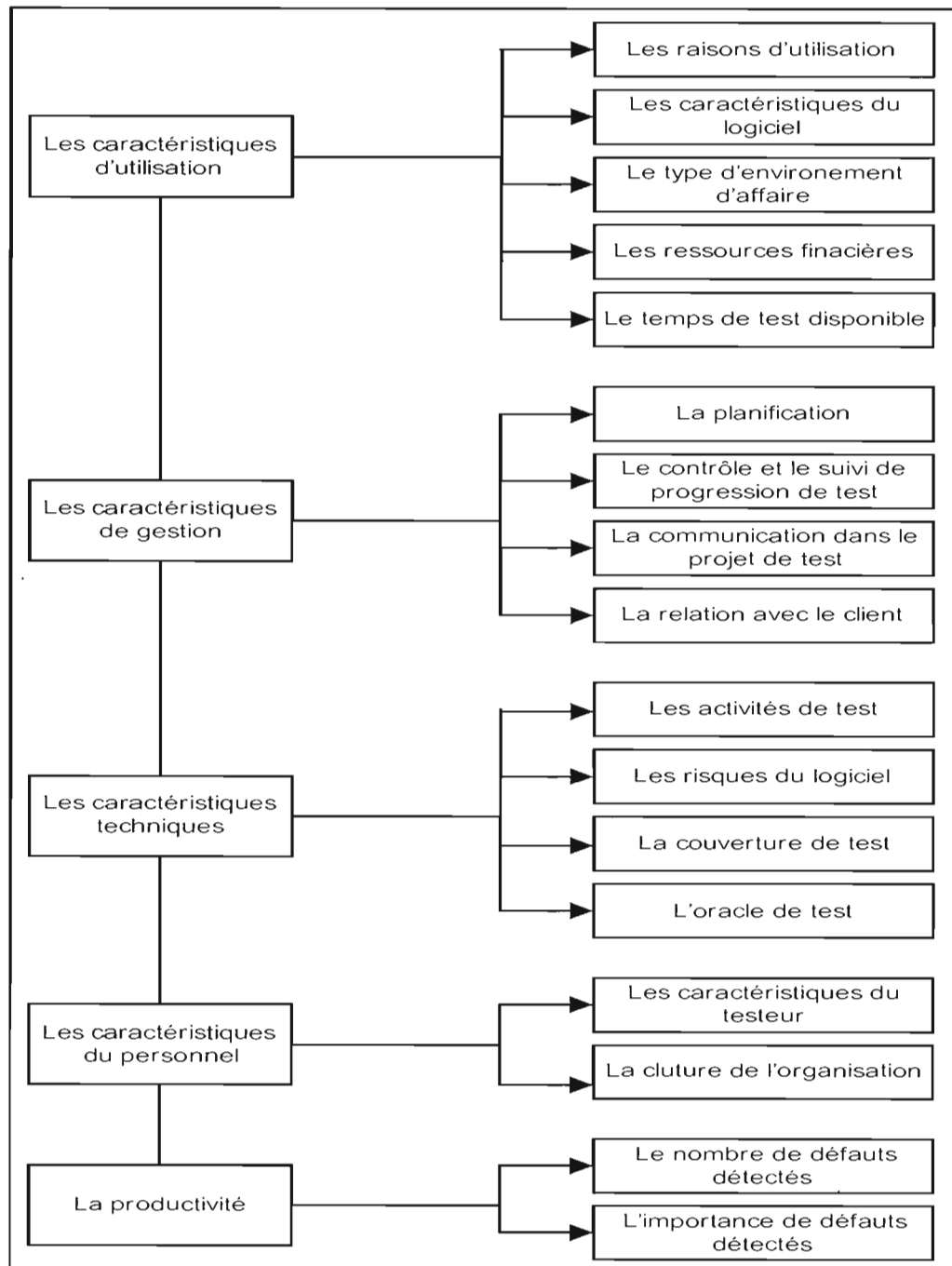
TE et le TS cette dimension de comparaison regroupe les facteurs suivants: les caractéristiques des testeurs et la culture de l'organisation où se déroule le projet de test.

6.2.5 La productivité

Nous avons ajouté cette dimension à notre cadre, vu l'importance de cet aspect dans notre travail de recherche. Ce critère constitue le centre de ce travail de recherche, à cause de la productivité du TE annoncée dans la littérature, surtout par les praticiens du CDS (Bach, 2003; Kaner, 2006). Les facteurs de cette dimension sont: le nombre de défauts détectés et l'importance de défauts détectés par chacune des deux approches de test, le TE et le TS. Ces facteurs de comparaison vont être traités de deux façons: théorique en se basant sur la littérature, et empirique ou expérimentale en se basant sur les résultats de notre étude empirique.

En résumé, notre cadre comparatif se compose des dimensions et des facteurs de comparaison illustrés sur la figure 6.1

Figure 6.1 : Le cadre conceptuel de comparaison



6.3 Conclusion

Au cours de ce chapitre, nous avons présenté un cadre conceptuel de comparaison. Nous avons identifié et défini dans ce cadre cinq dimensions principales de comparaison: les caractéristiques d'utilisation, les caractéristiques de gestion, les caractéristiques techniques, les caractéristiques du personnel, la productivité. Chacune de ces dimensions se décompose en plusieurs facteurs. Ces facteurs vont nous servir dans notre analyse comparative du TE et du TS qui sera abordée dans le chapitre qui suit.

CHAPITRE VII

ANALYSE COMPARATIVE DU TEST EXPLORATOIRE ET DU TEST SCÉNARISÉ

Dans ce chapitre, nous allons poursuivre notre discussion plus en détails sur les facteurs influençant l'adoption et l'adaptation de test exploratoire (TE) dans le cadre d'une analyse comparative des deux approches de test, le test exploratoire (TE) et le test scénarisé (TS). Nous comparons les deux approches en se basant sur les facteurs d'évaluation de notre cadre conceptuel de comparaison que nous avons développé dans le chapitre précédent. L'objectif de ce présent chapitre est d'identifier les contextes de test où le TE pourrait être utilisé en remplaçant la méthode habituelle scénarisée de test. En terminant, nous proposons un guide de sélection de l'approche de test. Ce guide récapitule les résultats de notre analyse comparative et tente de baliser une démarche d'analyse pouvant être intégrée à la réflexion stratégique des entreprises lors de choix de l'approche de test.

7.1 Comparaison selon les caractéristiques d'utilisation

Dans cette section, nous allons discuter les caractéristiques d'utilisation spécifiques pour chacune des deux approches de test, le TE et le TS. Tout d'abord, nous aborderons les raisons d'utilisation de chacune des deux approches de test. Puis, nous discutons l'influence des caractéristiques du logiciel à tester sur le choix de l'approche de test. Ces caractéristiques regroupent, la taille du logiciel, sa complexité et sa criticité. Ensuite, nous discuterons l'influence de type d'environnement d'affaire sur le choix de l'approche de test. Finalement, nous aborderons l'influence des facteurs, le temps de test disponible et les ressources financières sur le choix de l'approche de test.

7.1.1 Les raisons d'utilisation

La raison principale de l'utilisation du TE comme une approche primaire de test, était pour

s'accommoder à l'instabilité du logiciel développé et/ou l'absence des exigences du logiciel (Itkonen et Rautiainen, 2005; Petty, 2005; Lyndsay et van Eeden, 2003; Amland et Vaga, 2002). Ces études ont montré que le TE s'adapte à de telles situations. Cette adaptation s'explique par la possibilité d'utiliser le TE, sans avoir besoin d'utiliser les exigences du logiciel. En effet, le TE n'exige pas l'existence d'exigences formelles, du fait que les chartes de tests (le contenu de la mission de chaque session de test) pourraient être identifiées en utilisant n'importe quelle source d'information existante, mises à part les exigences du logiciel. Les auteurs Kaner et Bach (2004) ont décrit plusieurs références peuvent servir pendant l'identification des chartes. Souvent le manuel d'utilisateur est utilisé pour identifier ces chartes. Ces chartes identifient seulement les grandes lignes de la mission de test à accomplir. Par conséquent, l'introduction des changements sur le logiciel sous test ne pourrait introduire que des mises à jour minimales au pire des cas, comme l'ajout de nouvelles chartes correspondant aux nouvelles fonctionnalités ajoutées au logiciel.

Parmi les principes de l'école CDS, on retrouve que les projets se déroulent souvent d'une manière imprévisible. Ceci implique que l'approche de test devrait s'adapter à cette imprévisibilité. A cet effet, nous pouvons constater que l'objectif principal du TE est de s'adapter à l'imprévisibilité de projet de test, du fait qu'il est basé sur les principes de CDS.

La recherche menée par les auteurs Itkonen et Rautiainen (2005) a dévoilé d'autres raisons d'utilisation du TE à savoir : premièrement, la possibilité de tester le système du point de vue de l'utilisateur, autrement dit, tester la combinaison de plusieurs scénarios d'utilisation du logiciel. Deuxièmement, la difficulté de concevoir des cas de tests scénarisés détaillés à cause de l'interdépendance entre plusieurs unités logicielles. Troisièmement, la possibilité d'exploiter la créativité et l'expertise des testeurs dans la détection des défauts importants. Finalement, la limitation du temps de test et les ressources financières disponibles.

Par contre, les raisons d'utilisation de TS ne pourraient pas être dénombrées dans une liste déterminée de raisons d'utilisation. Du fait que, le TS constitue toujours la méthode habituelle et naturelle de test dans plusieurs entreprises. Cependant, nous avons constaté, d'après notre revue de littérature, que le TS ne s'accommode pas facilement aux changements

du logiciel. En effet, n'importe quel changement dans le logiciel nécessite la mise à jour de tous les artefacts de test déjà conçus, touchés par ce changement. L'effort nécessaire pour mettre à jour ces artefacts augmente proportionnellement avec l'augmentation de niveau de détail de ces artefacts. L'utilisation de standard de documentation IEEE 829 dans le TS nécessite, aussi un effort significatif pour mettre à jour les artefacts de test déjà conçus (Craig et Jaskiel, 2002 ; Petty, 2005). Notant, que le volume de modifications nécessaire accroît proportionnellement avec le volume de changements introduit sur le logiciel. Or, avec la culture rapide du développement du logiciel actuelle, les praticiens tendent souvent à commencer le développement du logiciel le plus tôt possible, avant que les exigences soient stables et mûries, afin de réduire le temps de mise en marché. Par la suite, la probabilité d'introduire des changements ultérieurs sur le logiciel est fort possible, parfois assez nombreux.

À cet effet, nous pouvons constater que la stabilité de projet du développement pourrait être l'une des raisons essentielles pour utiliser le TS. Notant que même le TE pourrait être utilisé dans ce type de projet. Dans une telle situation, d'autres facteurs pourront influencer le choix de l'approche convenable de test, à savoir, l'adoption de l'organisation du modèle d'amélioration de processus logiciel, comme le CMMI (Capability Maturity Model Integration). Dans ce genre de processus, la documentation et la mesure constituent des éléments fondamentaux. Par conséquent, le TS constitue le choix idéal dans ce type de projet de développement. Le domaine d'affaire de quelques types de projet du développement exige l'utilisation de TS. Autrement, la nécessité d'une documentation détaillée de l'activité de test exige l'utilisation de TS, comme les projets de test d'applications critiques par exemple. Par la suite, le besoin de documentation de processus de test pourrait être l'une des raisons pour utiliser le TS.

Nous concluons de cette discussion que le TE est plus approprié dans les projets de développement instables, grâce à sa grande capacité d'adaptation à l'imprévisibilité de projet de test. Tandis que le TS est plus approprié dans les projets de développement stables et qui ont besoin de documenter et mesurer l'activité de test.

7.1.2 Les caractéristiques du logiciel

7.1.2.1 La taille du logiciel

Il apparaît que le TE est plus approprié dans les petits et moyens projets de test. C'est à dire, le test des petites et moyennes applications. Cela, s'explique par l'effort nécessaire pour la gestion et le contrôle de TE. En effet, la gestion de TE nécessite que le responsable de test débriefe chaque testeur après l'exécution de chaque session de test, afin d'évaluer et d'approuver les résultats de la session. Or, ceci ne semblait pas être facile dans une grande équipe. Nous avons constaté ceci à travers les travaux de la littérature que nous avons présenté dans le chapitre IV. Alors, tous les logiciels qui ont été l'objet des études de cas étaient des petites applications développées par des petites ou moyennes entreprises (Itkonen et Rautiainen, 2005; Petty, 2005; Lyndsay et van Eeden, 2003; Amland et Vaga, 2002). Selon Lyndsay et van Eeden (2003), la collaboration et le partage des connaissances entre les membres de l'équipe de test peuvent améliorer la qualité de TE effectué. Cependant, nous croyons que la collaboration est plus facile entre les membres d'une petite équipe de test, qu'entre les membres d'une grande équipe.

Par contre, le TS est plus approprié dans les grands projets de test, associés à des grands projets du développement. En effet, dans ces projets, le projet de test constitue souvent un sous projet organisé et géré séparément du projet de développement du logiciel (Pyhajarvi et al., 2003). À cet effet, la planification et la documentation sont les moyens de gestion et de communication à l'intérieur et à l'extérieur de projet de test. S'ajoute à ceci, le fait que les grands projets puissent s'étaler sur plusieurs mois. Par la suite, la mémorisation et l'enregistrement des cas de test, sont nécessaires, afin de maintenir et tester l'application dans les phases ultérieures du développement d'une façon rentable (Beizer, 1990).

Nous concluons que le TE est plus approprié dans les petits et moyens projets de test, c'est-à-dire le test des petits et moyens logiciels. Tandis que le TS est plus approprié dans les grands projets de test, c'est-à-dire pour les grands logiciels.

7.1.2.2 La complexité du logiciel

La complexité du logiciel fait référence à la difficulté de compréhension et d'appréhension du logiciel. Autrement dit, la compréhension du logiciel n'est pas à la portée de tous les testeurs dans le projet. Par exemple, un logiciel qui implémente une nouvelle technologie. Alors, le TE ne semblait pas le meilleur choix dans cette situation, puisque l'apprentissage et l'appréhension du logiciel, nécessaires pour remplir l'activité de TE ne pourront pas être à la portée de tous les testeurs dans le projet de test. Le TS pourrait être la meilleure stratégie dans ce cas de test, du fait que les cas de tests pourraient être conçus par des experts dans la technologie implémentée et exécutés par des testeurs novices.

Il faut noter que parfois les logiciels complexes se développent en collaboration entre plusieurs compagnies (Kaner et al., 1999). Dans une telle situation le TS représente le bon choix, surtout s'il est documenté par le standard de la documentation IEEE 829. La documentation de l'activité de test permet de vérifier et d'inspecter formellement la qualité de test effectué au niveau de chaque entreprise participante dans le développement du logiciel. Le standard IEEE 829 peut introduire un protocole commun de communication entre les entreprises participantes dans le projet du développement (IEEE829, 1998).

Nous concluons que le TS est plus approprié dans les projets de test de logiciel complexe. Tandis que l'utilisation de TE dans tels projets dépend des compétences et de l'expertise des testeurs impliqués dans le projet.

7.1.2.3 La criticité du logiciel

En ce qui concerne le test des logiciels critiques, seulement le TS pourrait être utilisé. En effet, pour les logiciels critiques, comme les logiciels temps réel ou embarqués, seulement les méthodes scénarisées peuvent être utilisées. L'un des éléments essentiels de ces méthodes de tests est la documentation détaillée de l'activité de test. Cette documentation fera l'objet de plusieurs évaluations et inspections, afin de s'assurer de la qualité de l'activité de test effectuée. Dans certaines situations la documentation et l'activité de test devront suivre des normes et des standards spécifiques dans quelques domaines d'affaires. Nous avons constaté

ce fait à travers l'étude de cas de James et Wood (2003) présenté dans le chapitre IV. Les deux auteurs ont utilisé le TE comme une méthode complémentaire à l'approche scénarisée habituelle. Cette dernière devait suivre les normes de qualité du système (Quality System Regulation) dans le domaine de construction d'appareils médicaux.

Il faut rappeler que même Bach et al. (2002) ont signalé que les principes et les leçons présentés dans (Bach et al, 2002) de l'école de pensée Context Driven School (CDS) concernent les logiciels commerciaux seulement. À cet effet, nous croyons que le TE s'applique lui aussi à ce type de logiciels, du fait qu'il est basé sur les mêmes principes de l'école.

Nous concluons de cette discussion que seule le TS pourrait être utilisé dans le test des logiciels critiques.

7.1.3 Le type d'environnement d'affaire

Le type d'environnement d'affaire pourrait influencer le choix de l'approche de test, entre le TE et le TS. D'après notre revue de littérature, nous avons constaté que le TE s'adapte facilement aux changements du logiciel à tester. Par la suite, nous pouvons constater que le TE est plus approprié dans les environnements dynamiques. Le dynamisme fait référence au dynamisme de la technologie utilisée dans le développement du logiciel ou/et de la volatilité des besoins du client. À l'inverse le TS ne s'adapte pas facilement aux environnements dynamiques. De fait que la maintenance des artefacts de test nécessite du temps et des ressources financières considérables qui ne sont pas souvent disponibles dans la pratique du test, surtout dans les petites et moyennes entreprises. D'ailleurs, nous avons constaté cela à travers notre revue de littérature. La découverte et l'utilisation de TE ont été motivées dans la plupart des études de cas que nous avons présentées par l'incapacité de TS à répondre aux besoins des praticiens dans la limite du temps et des ressources disponibles (Petty, 2005; Amland et Vaga, 2002).

Le TS s'adapte très bien aux environnements stables. Dans ces environnements, les artefacts de test peuvent être spécifiés tôt dans le processus du développement, dans la limite des

ressources disponibles et aucun coût supplémentaire de changement n'est nécessaire. Le TS est plus approprié aussi dans d'autres types d'environnements d'affaires, comme les logiciels évolutifs et les logiciels développés sous contrat. Dans ce type d'environnements, le plan de test et les cas de tests devraient être livrés avec le logiciel. Parfois, le client détermine et négocie la structure, le format et le niveau de détail de ces artefacts dans le contrat. Il pourrait exiger l'utilisation de la norme de documentation IEEE 829 dans quelques situations (Kaner et al., 1999)

Les logiciels développés dans quelques domaines d'industrie exigent l'utilisation de TS, à cause de la nécessité d'une documentation détaillée de processus de test. Souvent, c'est le cas dans le test des logiciels qui peuvent causer des pertes importantes en cas de défaillance du logiciel dans le site de production. Alors, la documentation de test pourrait constituer un outil de défense dans les causes judiciaires (Kaner et al., 1999).

Nous pouvons conclure de cette analyse que le TE est plus approprié dans les environnements dynamiques. Par contre le TS est plus approprié dans les environnements stables, évolutifs, ou sous contrat et ceux qui nécessitent la documentation détaillée du processus de test.

7.1.4 Les ressources financières

Nous avons constaté d'après notre revue de littérature que les ressources financières disponibles dans le projet de test jouent un rôle important et crucial dans le choix de l'approche de test (Itkonen et Rautiainen, 2005; Petty, 2005 ; Lyndsay et van Eeden ,2003 ; Amland, Vaga, 2002)

Le TS nécessite des ressources financières importantes. La préparation, la conception des cas de tests et la documentation du processus de test, occasionnent des frais significatifs. En effet, l'effort nécessaire, en nombre de personnes/jours pendant le processus de test est plus grand en TS qu'en TE. Ce fait empêche l'utilisation de TS quand le budget de test est serré (Lyndsay, van Eeden, 2003). Contrairement, le TE ne nécessite pas une documentation rigoureuse pendant le processus de test, à part un investissement minime dans le processus d'identification des chartes de tests. La moindre coût de TE a été signalé par les praticiens qui

ont utilisé l'approche dans l'industrie de test (Petty, 2005; Lyndsay, Van Eeden, 2003; James, Wood, 2003).

Cependant, la différence dans les coûts entre le TE et le TS apparaît clairement lors de l'introduction des changements sur le produit sous test. Alors, la maintenance des artefacts de tests scénarisés a un coût supplémentaire, qui augmente en fonction du volume de changements introduits. Par contre, le changement du logiciel dans une activité de TE ne pourrait générer que quelques changements sur les chartes existantes tel que l'ajout des nouvelles chartes. Par la suite, le TE ne génère pas des coûts supplémentaires importants comme le TS. En fait, le coût moindre de TE était le principal motif de la découverte et de l'utilisation de cette approche par les professionnels et les praticiens dans l'industrie de test (Petty, 2005 ; Amland, Vaga, 2002).

Nous concluons de cette discussion que le TE est plus approprié dans les projets de test à ressources financières limitées. Tandis que le TS est plus approprié dans les projets de test qui ont des ressources financières importantes pour supporter ses frais.

7.1.5 Le temps de test disponible

Le temps disponible pour accomplir l'activité de test est l'un des facteurs essentiels pouvant influencer le choix de l'approche de test (Itkonen et Rautiainen, 2005; Petty, 2005 ; Lyndsay et van Eeden ,2003 ; Amland, Vaga, 2002). Le TS nécessite du temps considérable pour la planification et la conception des cas de tests avant le début de l'exécution physique des tests. Toutefois, avec la tendance préventive actuelle de test du logiciel, la planification et la conception de cas de tests scénarisés peuvent commencer dès le début de projet du développement, parallèlement avec l'implémentation du logiciel. À cet effet, il y a toujours suffisamment de temps pour planifier et préparer les cas de tests scénarisés. Le problème de temps se pose lors de l'introduction de changements sur le logiciel ultérieurement dans le cycle du développement, c'est-à-dire le changement de la conception du logiciel après l'élaboration des cas de tests associés à celle-ci. Dans ce cas, le temps nécessaire pour mettre à jour les artefacts de test déjà conçus et le temps disponible pour tester le logiciel pourrait avoir une incidence sur le déroulement normal de l'activité de test, qui peut changer

subtilement vers un test ad hoc ou dans la meilleure des cas au TE. En effet, la préparation des cas de tests tôt dans le processus n'est pas toujours fructueuse, parce que les exigences ne sont pas toujours stables au début du projet de développement. Par la suite la planification et la préparation des cas de tests en se basant sur ces spécifications changent souvent et génèrent des modifications et des mises à jour énormes. Aussi, l'emplacement des testeurs à la fin de la chaîne du développement c'est-à-dire sur le chemin critique de livraison du logiciel s'ajoute au problème que nous venons de citer. Alors, les testeurs accumulent les retards faits au début de la chaîne du développement, et se retrouvaient souvent dans des situations difficiles où ils doivent faire de compromis entre le temps disponible pour le test, la qualité attendue du logiciel et le budget. Dans de telles situations, souvent, les praticiens renoncent aux cas de tests scénarisés et s'en remettent au TE (Petty, 2005; Bach et al., 2002) Le TE leur permet d'investir le temps disponible dans le test du logiciel au lieu de mettre à jour les cas de tests scénarisés. Selon Bach et al. (2002), la préparation des cas de tests pendant la phase de conception du logiciel est une perte du temps, du fait que ces cas de tests pourraient être désuets ultérieurement dans le cycle du développement.

Par contre, le TE ne nécessite pas beaucoup du temps pour la préparation de chartes de test. Il suffit que le responsable de test prépare les chartes de tests en se basant sur n'importe quelle source d'information disponible (Kaner et Bach, 2004). Nous pouvons dire que c'est équivalent à la préparation d'un plan de test selon le patron IEEE 829, puisque l'objectif dans les deux cas est d'identifier ce qui doit être testé, les responsabilités, l'estimation de l'effort nécessaire pour remplir chaque mission dans le processus de test. Notons qu'avant l'introduction de la technique Session Based Exploratory Testing (SBET), Amland et Vaga (2002) ont utilisé un plan de test scénarisé pour identifier les chartes des sessions de tests exploratoires.

Nous concluons de cette discussion que le TE est plus approprié dans les projets de test où il y a peu de temps pour le test, surtout si les ressources financières ne permettent pas d'ajouter de personnel ou de faire d'autres contingences. Tandis que le TS est plus approprié dans les projets de test où il y a suffisamment de temps pour la préparation et la planification de test.

7.2 Comparaison selon les caractéristiques de gestion

Dans cette section, nous abordons les éléments principaux de gestion de l'activité de test dans les deux approches de test, le TS et le TE. Il s'agit de la planification de test, le contrôle et le suivi de la progression de test, la communication dans le projet de test et la relation avec le client.

7.2.1 La planification

Selon l'office québécois de la langue française, la planification est un ensemble de décisions ayant pour but d'établir un ordre d'application des actions avant leur exécution. Dans cette section nous allons discuter comment les deux approches abordent la planification en se basant sur cette définition.

En ce qui concerne le TS, l'activité de test est pilotée par le plan de test élaboré au début du projet de test. Ce plan situe les activités de test dans la stratégie globale de livraison du logiciel. La préparation de plan de test peut commencer au moment de la formulation des besoins et se développer et se raffiner pendant la phase de conception du logiciel. Cela s'inclut dans le cadre de test préventif. Ce type de test dicte que la planification de TS peut prévenir les erreurs dans la phase de conception avant que celles-ci se transforment à des défauts dans le code (Craig et Jaskiel, 2002; Perry, 2002; Swebok, 2004). Alors, du processus de planification résulte le plan de test. Ce plan décrit les items et les caractéristiques (fonctionnalité, performance, utilisabilité, etc.) qui devraient être testés, les caractéristiques qui ne devraient pas être testées, les responsabilités, les livrables, les besoins environnementaux et l'échéancier du projet de test. Par la suite, tous les détails des tests sont identifiés et rigoureusement planifiés avant le début de leur exécution. En fait, le plan de test dans une activité de TS vise deux objectifs essentiels: le premier, la planification de l'activité de test, le deuxième, l'établissement des canaux de communications entre les intervenants à l'intérieur et l'extérieur du projet de test. Selon (IEEE 829, 1998) l'utilisation de standard IEEE 829 pourrait améliorer la qualité de ces canaux de communication.

Le TE introduit lui aussi la planification, mais avec un rôle et une signification différents. Au début de l'activité de test, le responsable de test identifie une liste des items à tester. Ces

items constituent les chartes de tests (déjà évoqué dans le chapitre III). Cependant, il ne s'agit pas d'une identification complète de toutes les chartes avant le début de l'exécution de test, mais d'une planification préliminaire à quelques chartes de test, en se basant sur les informations disponibles au début du projet de test. Ce plan s'améliore pendant le déroulement du projet de test, grâce aux notes résultant de l'exécution de chartes de test. Autrement, d'autres chartes de test s'ajoutent au fur et à mesure que les chartes préliminaires s'exécutent. En effet, pendant l'exécution de charte de test, le testeur peut explorer n'importe quel partie du logiciel et rapporter ses constations et ses observations dans les notes de session, ce que Jonathan Bach (2000) appelle «Opportunité». Ces notes font l'objet d'une évaluation avec le responsable de test. Suite à cette évaluation le responsable de test pourrait ajouter des nouvelles chartes correspondant et mettre à jour le plan de chartes. Selon James et Wood (2003) la planification de TE est une planification continue (Figure3.2). Dans la même perspective, Copeland (2003) classifie la planification de TE comme une planification adaptative qui se change à chaque moment à la venue de nouvelles informations pendant le processus de test, alors que la planification scénarisée est une planification classique qui identifie toutes les actions et les détails de test avant l'exécution de ces actions.

Nous pouvons constater que la planification dans une activité de TE constitue un moyen de contrôle et d'encadrement de processus. En fait, il ne s'agit pas d'une planification telle qu'elle est définie par le dictionnaire où toutes les actions devraient être spécifiées avant leur exécution. Mais c'est une planification introduite par Jonathan Bach (2000) pour gérer l'activité de test et introduire le contrôle et la mesure sur le TE libre. Donc, il s'agit d'une planification qui vise l'exécution de la mission de test du logiciel plutôt que la documentation et l'archivage en texte. Nous pouvons constater aussi que la planification de TS est plus certaine et rigoureuse que la planification de TE. En effet, la planification de TS tend à préciser tous les détails fins du processus de test avant le début de leur exécution, tandis que le TE tend à préciser l'essentiel du processus de test et compte sur l'exploration des testeurs pendant l'exécution des sessions de test pour raffiner et améliorer le plan initial.

Nous concluons de cette discussion que le TS est plus approprié dans les projets de test qui nécessitent une planification certaine et rigoureuse de l'activité de test. Par exemple, les

situations où la plate-forme de test est disponible seulement par intermittence. Comme un projet client serveur où les serveurs configurés devraient être partagés par l'activité de test et le développement. La logistique d'une telle situation peut exiger l'utilisation de TS qui pourrait fournir une planification rigoureuse de l'activité de test pour profiter au maximum du temps limité pour les tests (Bach, 2003). Par contre, le TE est plus approprié dans les projets de test flexibles.

7.2.2 Le contrôle et le suivi de progression de test

L'objectif de contrôle et du suivi du test est de fournir un retour et une visibilité sur les activités de test. Les informations à suivre peuvent être recueillies manuellement ou automatiquement et peuvent être utilisées pour évaluer les critères de sorties, comme la couverture de test. Des mesures peuvent aussi être utilisées pour évaluer l'avancement par rapport au calendrier et au budget planifiés.

Selon Craig et Jaskiel (2002) le contrôle et le suivi du test sont parmi les problèmes auxquels le responsable devrait faire face dans un projet de test. Alors, le responsable devrait trouver une méthode pour retracer ou suivre le déroulement de l'activité de test. Par la suite, il devrait être capable de faire un rapport sur le statut des tests à n'importe quel moment tant que le produit est toujours sous test. Il devrait aussi pouvoir répondre aux questions typiques qui se posent régulièrement pendant les tests:

- Comment se déroulent les tests?
- Quel est le pourcentage des tests accomplis?
- Quel est le pourcentage des tests réussis ?

Le TS se prête très bien à la mesure. Un nombre total de cas de tests peut être calculé, et une métrique de test peut être créée, mesurant par exemple le pourcentage des cas de tests qui ont été exécutés. Des méthodes spécifiques proposées dans la littérature permettent de mesurer la progression de l'activité de TS et de prévoir la date de livraison du logiciel en se basant sur le nombre de cas de tests restants avant la complétude de l'activité de test (Kan, 2001; Craig et Jaskiel, 2002)

Quant au TE, les mesures collectées pendant le test et le débriefing permettent d'estimer la productivité ou le rendement de chaque membre de l'équipe de test pendant le projet de test en cours. Cela, avec le nombre de sessions complétées, pourrait aider dans l'estimation de la quantité du travail restant avant la fin du cycle de test (Jonathan Bach, 2000). Notons que le sujet du contrôle de la progression n'est pas toujours bien abordé dans la littérature, à part ce mécanisme proposé par Jonathan Bach (2000). Toutefois, ce mécanisme aide seulement dans l'élaboration d'une estimation de la quantité du travail restant avant la fin du cycle de test. Il ne s'agit que d'une estimation, la prévision se basant sur cette estimation étant incertaine.

Nous pouvons conclure de cette discussion que le contrôle et le suivi de la progression de test dans le TS est mesurable, alors qu'il n'est qu'un estimé. En conséquence, nous concluons que le TS est plus approprié dans les projets de tests qui ont un échéancier fixe, alors que le TE est approprié dans les projets de test qui ont un échéancier flexible et tolérant au retard qui pourrait résulter d'une mauvaise estimation.

7.2.3 La communication dans le projet de test

Le TE mise fortement sur les connaissances tacites et sur les compétences interpersonnelles. Comme nous avons déjà vu dans la revue de littérature, le TE est basé sur les connaissances tacites du testeur et son expertise (Itkonen et Rautiainen, 2005; Petty, 2005; Lyndsay et van Eeden, 2003; Wood et James, 2003). Les compétences interpersonnelles jouent aussi un rôle important dans la qualité du travail (Lyndsay et van Eeden, 2003), la communication étant essentielle dans le TE. Nous avons constaté d'après notre revue de littérature que la communication et l'échange des connaissances interviennent de différentes façons dans le TE. La première est le partage des connaissances entre le testeur et d'autres intervenants hors de l'équipe de test comme le client. En effet, le testeur chargé de l'exécution d'une mission de TE pourrait collecter les informations nécessaires sur le logiciel à tester à partir des réunions ou des discussions avec différents intervenants dans le projet de développement du logiciel à tester, afin de comprendre et d'apprendre le logiciel et ses caractéristiques fonctionnelles (Kaner et Bach, 2004).

La deuxième forme de communication est entre testeurs et responsable du test. Ce dernier devrait débriefing chaque testeur après l'exécution de chacune des chartes de test, pour évaluer

le travail accompli. Alors pendant le débriefing le responsable pourrait échanger ses connaissances avec le testeur pour le guider (Jonathan Bach, 2004; Lyndsay et van Eeden, 2003). La communication pourrait aussi prendre la forme d'une collaboration entre les testeurs. La communication entre les membres de l'équipe de test pourrait améliorer la qualité de TE effectué. En effet, de bons canaux de communication permettent d'échanger les expériences et les connaissances dans l'équipe de façon à ce que les membres plus expérimentés aident et encadrent les membres moins expérimentés (Lyndsay et van Eeden, 2003).

La troisième forme d'échanges des connaissances peut apparaître pendant la session de test entre deux personnes qui se chargent de l'exécution de la même mission de test, définie sous le terme de «test par paires». Nous avons noté cela à travers les études de cas que nous avons proposé dans la revue de littérature, parfois entre deux testeurs (Amand et Vaga, 2002) ou entre un testeur et un développeur (Petty, 2005).

Le TS, au contraire, se mise beaucoup sur les connaissances explicitement documentées. La communication entre les praticiens dans le projet de test est centrée autour des livrables bien déterminés. En effet, souvent les testeurs chargés de la conception (testeurs seniors) communiquent avec les testeurs chargés de l'exécution des tests (testeurs juniors) par les procédures de test. Ces derniers communiquent réciproquement par les rapports d'incidents. En ce qui concerne la communication entre les testeurs et les développeurs, dans la plupart des cas, la communication se fait à travers «le Bug Tracking System », le système qui enregistre les défauts détectés. Nous notons que dans quelques situations, il règne une relation difficile entre les développeurs et les testeurs (Bach et al., 2002), du fait que les développeurs sentent que les testeurs détruisent leur travail et se sentent responsables des défaillances détectées.

Nous pouvons conclure que la gestion de TE se fonde sur la communication entre les intervenants dans le projet de test, même la qualité des tests effectués dépend de la qualité de communication entre le responsable des tests et les testeurs, tandis que la gestion dans le TS se fonde sur la documentation.

En conséquence, il semble que les contextes supportant la communication informelle sont plus appropriés et favorables au TE.

7.2.4 La relation avec le client

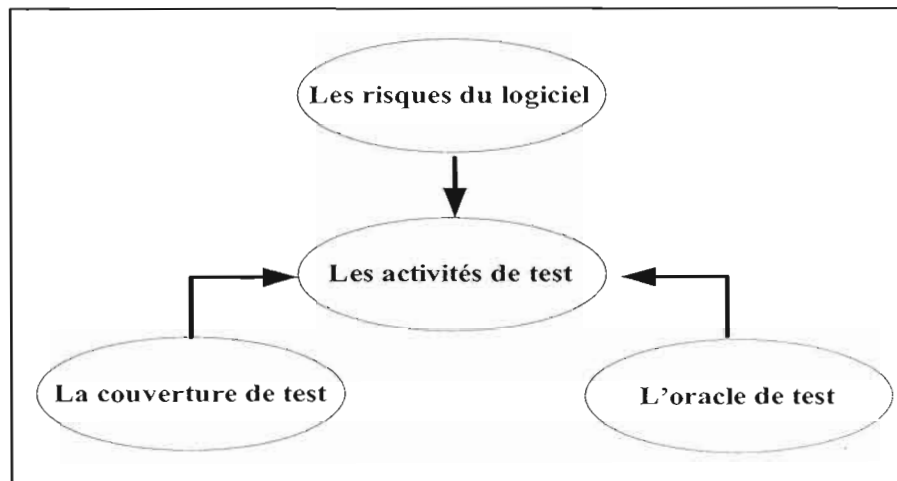
Selon Kaner et Bach (2004) la conversation avec le client pourrait être une source d'information importante pour la compréhension et l'appréhension du logiciel dans une activité de TE. À cet effet, une bonne relation avec le client est essentielle dans le TE. Par contre cette relation n'est pas nécessaire dans une activité de TS, du fait que les cas de test sont conçus à partir des exigences du système. Cependant, il est faux de dire que cette relation entre le testeur et le client n'est pas souhaitée dans le TS. Selon Craig et Jaskiel (2002) le client peut jouer un rôle important dans l'activité de TS par la participation dans les réunions d'identification de la liste des risques du logiciel. Ceci pourrait optimiser et orienter l'activité de test vers les risques importants du logiciel.

Nous concluons de cette discussion qu'une bonne relation avec le client est essentielle dans le TE et pourrait améliorer la compréhension du logiciel. Par contre elle est souhaitée et non obligatoire dans le TS.

7.3 Comparaison selon les caractéristiques techniques

Nous allons discuter dans cette section des caractéristiques techniques de deux approches de test le TE et le TS. Ces caractéristiques portent sur le processus de test, les activités de test, les artefacts créés et les éléments nécessaires pour le bon déroulement de test. Nous allons essayer de déceler comment les deux approches de test, le TE et le TS, abordent chaque étape de l'activité de test. Selon (Swebok, 2004; Pyhajarvi et al., 2003; Craig et Jaskiel, 2002 ; Perry, 2000), les activités de test comportent les étapes suivantes: la planification, la conception de tests et l'exécution de tests. Toutefois, ces activités sont influencées selon (Bolton, 2005) par trois facteurs, à savoir: les risques du logiciel, l'oracle de test et la couverture de test, tel qu'illustré sur la figure 7.1

Figure 7.1 : Les éléments essentiels du test du logiciel (Bolton, Kaner et Bach, 2005)



7.3.1 Les activités de test

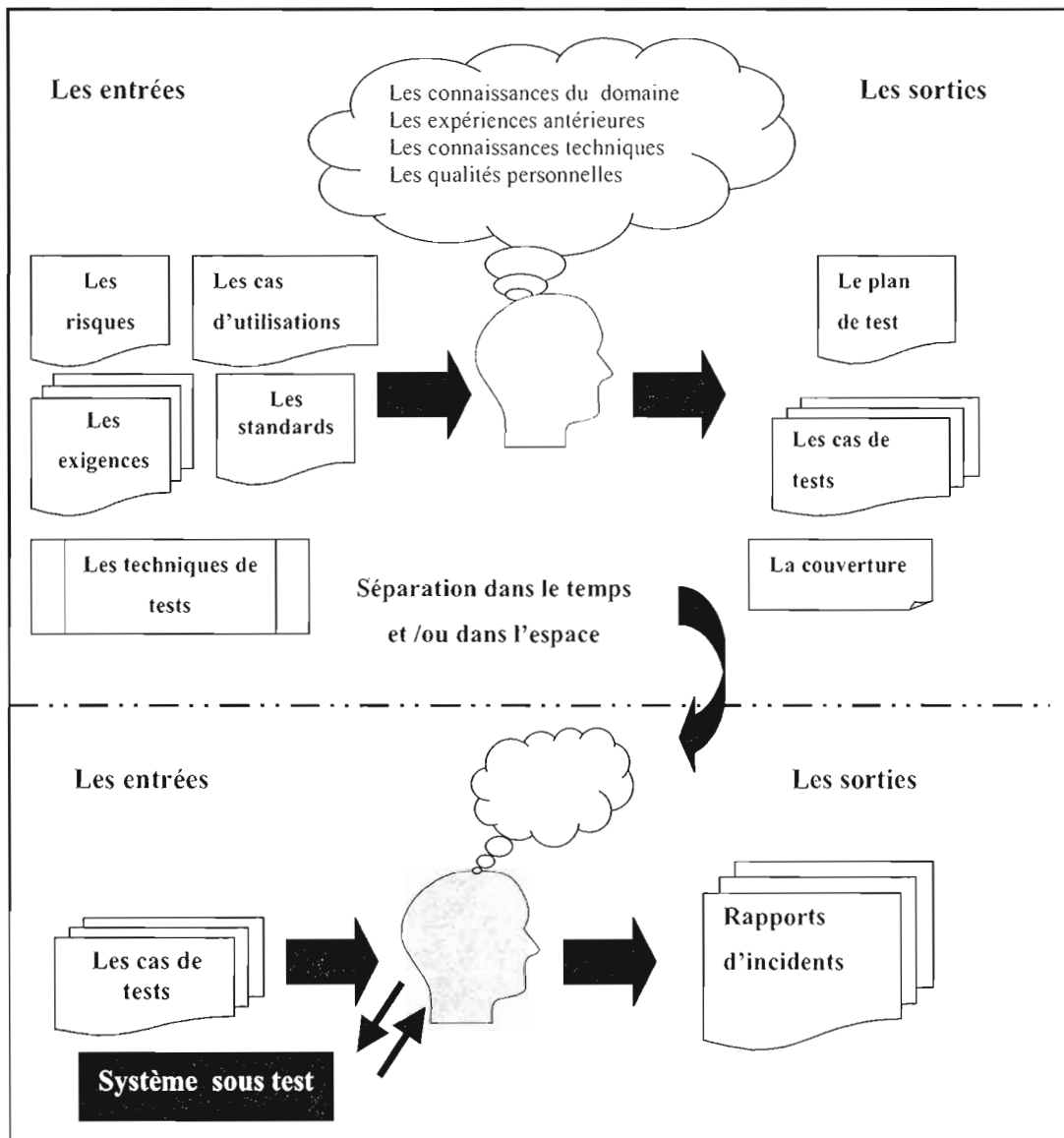
En ce qui concerne le TS, les cas de test sont conçus au début de l'activité de test, principalement à partir des exigences du logiciel. Chaque cas de test devrait être sous le contrôle de la gestion de configuration du logiciel et inclure les résultats prévus de chaque test (Swebok, 2004; Perry, 2000; Hetzel, 1988). La conception et la planification pourraient commencer en parallèle avec le projet de développement.

Tel qu'illustré à la figure 7.2, la préparation de plan de test et des cas de test se font par le testeur, souvent un testeur senior, en utilisant les entrées spécifiques de projet de test en cours. Alors ces entrées peuvent inclure les exigences du logiciel, les standards et les normes à respecter qui varient selon le domaine de test. Dans certaines situations, le test peut être guidé par divers objectifs, comme les risques du logiciel ou les cas d'utilisations, pour identifier les priorités de test et focaliser sa stratégie (Swebok, 2004). Quant aux techniques de test, elles sont choisies selon la nature du logiciel sous test, l'efficacité et la précision souhaitées (Craig et Jaskiel, 2003; Biezer, 1990).

S'ajoutent à ces entrées citées ci dessus, les compétences et les qualifications du testeur senior chargé de la conception de tests : ses connaissances, comme les connaissances du

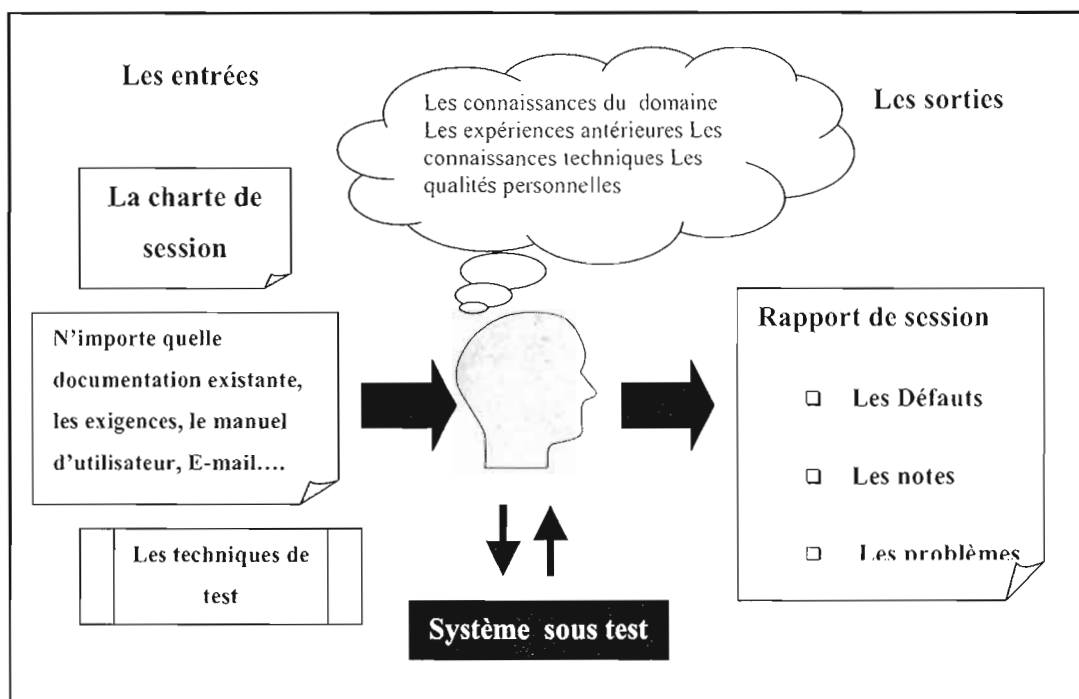
domaine du logiciel à tester, ses expériences antérieures, ses connaissances techniques et ses qualités personnelles. L'interaction de toutes les entrées permet d'identifier le plan et les cas de test. Le pourcentage de couverture de test atteint pourrait être mesuré à ce niveau du processus de test, à l'aide de la matrice de traçabilité figure 2.1

Figure 7.2 : Les activités de test scénarisé (Adapté et traduit de Bach, 2006)



Les cas de tests sont souvent exécutés ultérieurement par des testeurs juniors quand le logiciel devient disponible pour le test. Ces testeurs se chargent de l'exécution des procédures de test. Ces dernières décrivent tous les détails fins nécessaires à l'exécution des tests, incluant les résultats prévus, comme nous l'avons déjà proposé dans le chapitre II. Il suffit que les testeurs comparent les résultats prévus et les résultats observés. En cas d'apparition d'une défaillance, le testeur rapporte les résultats dans le rapport d'incident prévu à cet effet selon la norme IEEE 829.

Figure 7.3 : Les activités de test exploratoire (Adapté et traduit de Bach, 2006)



Par contre, tel qu'illustré à la figure 7.3 les cas de tests de TE sont conçus pendant le test. En effet, le testeur reçoit en entrée la charte de la session de test, qui identifie les grandes lignes de sa mission de test. Il doit s'informer sur le logiciel en utilisant n'importe quelle source d'information existante dans le projet de test en cours, comme le manuel d'utilisateur, des discussions avec le client et les développeurs et d'autres. Par la suite, il doit identifier les techniques convenables de test. Parfois ces techniques de test peuvent être mentionnées dans la charte de test, selon la complexité et le risque d'items à tester traités dans la charte. Alors,

pendant l'exécution de sa mission le testeur apprend, explore conçoit et exécute les tests (Bach, 2003). Chaque cas de test bénéficie de l'information obtenue de tests antérieurs et la maturité des connaissances accumulées à chaque moment de l'activité de test (Bach et al., 2002). Les résultats de la session de test sont rapportés dans le rapport de la session. Ce rapport fera l'objet d'un débriefing avec le responsable de test, afin de valider le travail effectué par le testeur.

Nous concluons de cette discussion que le TS est plus approprié dans les projets de test favorisant la répétitivité et l'auditabilité de tests. Par contre, le TE est plus approprié dans les projets de test favorisant la créativité et l'adaptabilité à chaque moment de processus de test.

7.3.2 Les risques du logiciel

Selon le dictionnaire de l'office québécois de la langue française, le risque informatique est la probabilité plus au moins grande de voir une menace informatique se transformer en événement réel entraînant une perte. Il se mesure à la fois par la probabilité d'occurrence d'une menace et par l'impact de sa réalisation. Dans la littérature de test du logiciel, le risque du logiciel se définit comme la probabilité d'occurrence et l'impact de la réalisation d'une défaillance dans le logiciel (Craig et Jaskiel, 2002; Lyndsay et van Eeden, 2003).

L'impossibilité de test complet du logiciel (Pressman, 1997; Kaner, 1997; Hetzel, 1988) et les cycles de développement de plus en plus courts, ont poussé les intervenants dans le domaine de test du logiciel à chercher des techniques leur permettant d'exploiter le temps consacré au test, d'une façon optimale, telle que l'analyse de risque du logiciel. Selon Swebok (2004), les différentes étapes de tests pourraient être guidées par les risques associés au logiciel pour identifier sa priorité et focaliser sa stratégie. Ces risques sont identifiés par le biais d'une analyse qui pourrait être faite pendant la phase de préparation de l'activité de test. Cette analyse permet de déterminer les éléments du logiciel les plus susceptibles de contenir le plus de défauts. Les résultats de cette activité sont utilisés pendant le planning pour déterminer la stratégie de test, les priorités et la profondeur de test nécessaire (Craig et Jaskiel, 2002; Lyndsay et van Eeden, 2003; Perry, 2000).

Le standard de documentation IEEE 829 a identifié une section dans le patron de plan de test

pour les risques et contingences. En fait, cette section n'identifie que les risques pouvant empêcher le déroulement normal de plan de test. Dans la pratique de test, les entreprises divisent cette clause de plan de test a deux clauses: une, traite les risques pouvant empêcher l'exécution normale de plan de test ainsi les contingences convenables pour les surmonter, l'autre traite et identifie les risques du logiciel (Craig et Jaskiel, 2002) et cite les procédures à entreprendre dans le test pour minimiser ses probabilités d'occurrence et ses impacts en cas de réalisation.

Dans une activité de TS, l'analyse de risques associés au logiciel se fait à partir de documents des exigences et de conception du logiciel. Le livrable de cette phase est une matrice de risque montrant le degré de risque acceptable pour chaque fonction ou secteur du logiciel et sa criticité aux affaires (Craig et Jaskiel, 2002). Par la suite, l'approche de test et l'effort de test nécessaire de chaque élément de cette matrice pourront être fixés et documentés selon le degré de risque calculé. Ainsi, les cas de test conçus pourront être revus et inspectés, par plusieurs intervenants dans le projet de test, autant de fois que nécessaire pour s'assurer de la profondeur et la couverture de tests des zones risquées du logiciel.

Selon (Bach, 2003; Kaner et Tinkham, 2003) le TE pourrait être guidé par une liste de risques. Le responsable de test identifie cette liste en se basant sur n'importe quelle référence ou source d'information existant dans le projet de test (Jonathan Bach, 2004; Lyndsay et van Eeden, 2003). Suite à cette identification, les chartes correspondant aux éléments de cette liste seront plus détaillées que les autres chartes, et pourront même décrire les techniques à utiliser pendant le test (Jonathan Bach, 2004). Cependant, le degré au bas niveau pour lequel chaque élément de la liste est testé ne pourrait pas être assuré, même avec les notes de session et le débriefing avec le responsable. Par conséquent le TE porte le risque que certaines parties de grands risques ne soient pas testées correctement.

Nous concluons de cette discussion que l'auditabilité de TS assure que les risques du logiciel soient bien testés, par conséquent le TS pourrait être plus approprié dans les projets de test présentent un niveau élevé de risque. Tandis que le TE ne garantit pas que les secteurs à risque soient bien couverts.

7.3.3 La couverture de test

La couverture de test est la mesure de ce qui a été testé proportionnellement à ce qui pourrait être testé (Kaner, 1996). C'est le degré, exprimé en pourcentage, selon lequel un élément de couverture (les exigences, lignes de code, branches, etc.) a été exécuté lors d'une suite de tests. C'est une métrique importante dans l'activité de test logiciel. Sa pertinence réside dans le fait qu'il permet d'évaluer la qualité des tests effectués et de savoir si le logiciel a subi assez de tests (Swebok, 2004). Pratiquement, la mesure de la couverture des exigences et de la couverture du code sont les deux formes de mesures de couverture les plus utilisées et discutées dans la littérature et les plus supportées par les outils dans le domaine de test du logiciel.

La matrice de traçabilité (Figure 2.1) constitue la première forme de mesure de couverture en type de test boîte noire. En effet, cette matrice montre à quel degré chaque exigence ou caractéristique du logiciel (la fiabilité, l'utilisabilité, etc.) a été testée, en illustrant le nombre de cas de tests qui la couvre (Craig et Jaskiel, 2002; Bach et al, 2002). Puis, des inspections et des revues formelles des cas de tests permettent d'évaluer la qualité des cas de tests conçus et de s'assurer que les secteurs identifiés pendant l'analyse de risque du logiciel sont bien couverts par les tests (Craig et Jaskiel, 2002)

La deuxième forme est la mesure de couverture de code. C'est une méthode d'analyse qui détermine quelles parties du programme ont été exécutées (couvertes) par une suite de tests et quelles parties ne l'ont pas été, par exemple la couverture des lignes de code, des décisions ou des conditions. Dans le type de test de type boîte noire, la couverture de lignes de code s'utilise souvent en utilisant un logiciel outil, appelé «moniteur de test». Ce moniteur mesure ou calcule le pourcentage de lignes de code exécuté lors de l'exécution des cas de tests scénarisés (Kaner et al, 1999 ; Marick, 1997). Alors, si le pourcentage mesuré est insuffisant, des cas de tests peuvent être ajoutés, afin d'atteindre le pourcentage visé.

Bach (2003) a mentionné que les chartes de tests peuvent être identifiées selon une liste ou matrice de couverture, c'est-à-dire une liste des éléments du logiciel à recouvrir dans le test. Cependant, la couverture de bas niveau, ne pouvait pas être mesurée du fait que les méthodes

traditionnelles que nous venons de citer dans le TS ne pourraient pas être utilisées dans le TE. En conséquence, la couverture du test n'est pas claire, ni mesurable dans le TE. Les auteurs Itkonen et Rautiainen (2005) ont mentionné que ce manque de mesure de couverture est la plus grande lacune du TE. Lindsay et van Eeden (2003) ont proposé une technique pour mesurer la couverture de test que nous avons décrite dans la revue de littérature. Quoique la technique soit innovatrice, son succès dépend aussi de l'expertise de testeur et de sa capacité de faire un bon jugement sur le pourcentage couvert par les tests pendant la session de test.

D'un autre point de vue, la mesure de couverture est très utile pour la prise de décision de l'arrêt de test. En effet, une des choses les plus difficiles dans le test de logiciel est de savoir quand le logiciel est suffisamment testé et prêt à être livré. Évidemment, le logiciel est bien testé quand tous les bogues sont trouvés. Cependant, impossible de savoir s'ils sont tous trouvés. Ce problème a été reconnu par Dijkstra en 1972 par sa citation célèbre qui déclare que «le test du programme peut être utilisé pour montrer la présence des bogues, mais jamais pour montrer leur absence». De ce fait, le recours aux critères permettant la prise de décision de l'arrêt de tests reste nécessaire. Selon Copeland (2004), les critères les plus utilisés dans la pratique sont: le budget, le calendrier de test et la couverture de tests. En conséquence, la couverture fournit un appui utile pour la prise de décision de l'arrêt de tests (Swebok, 2004). Toutefois, les praticiens dans l'industrie de test qui ont utilisé le TE comme une méthodologie primaire de test et les auteurs fondateurs de l'approche (Bach et al., 2002) n'ont pas présenté les mécanismes et les techniques qu'ils ont utilisés pour prendre la décision d'arrêter le test.

Nous concluons de cette discussion que la mesure de couverture de test ne peut pas être mesurée dans le TE. Par conséquent le TE n'est pas approprié dans les projets de test qui nécessitent que la couverture de test soit mesurée.

7.3.4 L'oracle de test

Selon Hoffman (1999) le test du logiciel est la comparaison du comportement du logiciel avec le comportement attendu de celui-ci. Ce comportement attendu est connu sous le terme «Oracle».

Nous allons aborder dans cette section les façons d'élaboration de l'oracle de test dans les deux approches de test, le TS et le TE.

Selon Swabok (2004) un oracle de test est n'importe quel agent humain ou mécanique qui statue si un programme s'est comporté correctement dans un test donné, et produit en conséquence un verdict de passage ou échec de test. Selon Biezer (1990) un oracle de test est n'importe quel programme, processus ou ensemble de données qui spécifient les résultats prévus.

Dans une approche scénarisée, l'évaluation de comportement du logiciel sous test est déjà intrinsèque aux cas des tests qui mentionnent les résultats prévus. Ces résultats servent comme un oracle et guident le testeur pendant le test. Cet oracle de test est de type entrée/sortie (Biezer, 1990), c'est-à-dire le testeur devrait exécuter le logiciel en utilisant les entrées spécifiées dans le cas de test, puis comparer les résultats observés aux sorties spécifiées dans le cas de test. S'ils sont semblables, le logiciel passe le test, sinon, il échoue le test.

Selon Bach (1999) l'oracle de test est une stratégie élaborée au moment du test pour déterminer si un comportement observé du logiciel est correct ou non. Alors, nous pouvons constater de cette définition que l'oracle de test dans le TE se construit en temps réel pendant le test. Le testeur formule une idée sur le comportement normal et correct du logiciel en se basant sur ses intuitions et anticipations, ses compétences et sa compréhension du logiciel à tester.

Afin de faciliter et guider la réflexion de testeur exploratoire pendant la session de test, Bach et Kaner (2005) ont proposé une liste des heuristiques. Ces dernières pourraient aider le testeur à construire l'oracle de test, en se basant sur la vérification de la cohérence selon plusieurs dimensions. Chacune de ces dimensions exploite un aspect particulier de contexte de projet de test, pour savoir si le comportement observé après l'exécution d'un test donné est normal ou non, par la suite, prendre une décision de passage ou d'échec de test.

Cette liste inclut :

- Cohérence du logiciel: le comportement de chaque fonction est cohérent avec le comportement de d'autres fonctions comparables du logiciel ou avec le pattern fonctionnel;
- Cohérence par rapport aux logiciels comparables: le comportement de chaque fonction du logiciel est cohérent avec le comportement d'autres fonctions de logiciels comparables;
- Cohérence avec l'historique: le comportement actuel du logiciel est cohérent avec son comportement antérieur;
- Cohérence avec l'image de l'organisation: le comportement du logiciel est cohérent avec l'image que l'organisation voulait projeter;
- Cohérence avec les exigences : le comportement est cohérent avec la documentation existante et ce qui est annoncé sur le produit;
- Cohérence avec les spécifications et les réglementations: le comportement est cohérent avec les exigences et les normes qui devaient être respectées dans le projet de test;
- Cohérence avec les attentes de l'utilisateur: le comportement est cohérent avec ce que l'utilisateur attend du logiciel;
- Cohérence avec l'objectif du produit: le comportement est cohérent avec le but apparent et la fonction globale du produit;

Dans la même perspective, Whittaker (2003) a proposé un modèle intitulé « Fault Model » pour guider le testeur dans l'élaboration de l'oracle de test. L'idée principale de ce modèle est que le logiciel a quatre capacités principales: accepter les entrées, enregistrer les données, traiter les données entrées et enregistrées et produire des sorties. Donc, si le logiciel ne fait pas l'une de ces étapes correctement pendant l'exécution d'un cas de test, il échoue le test. Ce modèle fait partie des techniques citées et recommandées par Bach et Kaner (2004). Ces

mécanismes s'ajoutent à d'autres qui visent la simplification de la mission de testeur pendant la session de test, ainsi que l'amélioration des résultats de TE. Mais une question qui se pose: est-ce que n'importe quel testeur pourrait être capable d'utiliser toutes ces techniques? Il semblerait que ces techniques nécessitent un testeur compétent et expérimenté.

D'un autre point de vue, selon Kaner (2004), la spécification des résultats prévus de chaque test dans le TS pourrait avoir des conséquences négatives sur l'efficacité de l'activité de test. Selon cet auteur, le logiciel ou le système informatique pourrait échouer de plusieurs façons imprévues. Par conséquent, l'oracle entrée/sortie de TS pourrait desservir au lieu de servir dans le test, parce qu'il peut empêcher la détection des défauts non prévus dans le cas de test.

Nous concluons de cette discussion que le TS permet d'avoir le même oracle de test chaque fois que les cas de test sont exécutés. Il permet aussi de vérifier le logiciel formellement par rapport ses spécifications. Par contre, l'oracle de TE est variable et permet de comparer le logiciel contre les prévisions du testeur.

7.4 Comparaison selon les caractéristiques du personnel

Dans cette section, nous allons discuter de l'influence des caractéristiques du personnel sur le choix de l'approche de test. Les caractéristiques du personnel regroupent deux facteurs: les caractéristiques du testeur et la culture de l'organisation où se déroule le projet de test.

7.4.1 Les caractéristiques du testeur

En général, tel qu'illustré sur les figures 7.2 et 7.3, les deux approches le TE et le TS nécessitent des qualifications et des compétences pour que le test soit efficace et atteigne ses objectifs. La différence réside dans le niveau d'application de ces compétences. En effet, dans une activité de TS n'importe quel testeur peut exécuter les procédures de tests. Ces procédures sont décomposées en étapes claires et faciles à suivre, comme nous l'avons déjà évoqué dans le chapitre II. Par conséquent, l'exécution des tests scénarisés n'exige pas des testeurs très expérimentés et fortement habiles et même un testeur qui vient juste de commencer sa carrière en test logiciel pourrait faire face (Craig, Jaskiel, 2002).

Par contre, la conception des cas de tests, exige la compétence. Donc, c'est à ce niveau que

les qualifications et l'expertise sont nécessaires. Parce qu'il y a un délai entre la création des cas de tests et leur exécution, différents testeurs peuvent concevoir et exécuter les cas de tests. De ce fait, les tests peuvent être conçus par des testeurs compétents ou même par un seul testeur expert et être délégués à plusieurs moins expérimentés. Bref, il n'y a aucun besoin d'avoir seulement des testeurs experts dans une équipe de test scénarisé.

Par contre, le TE nécessite des compétences et de qualifications importantes pour la conception et l'exécution des tests (Itkonen et Rautiainen, 2005; Petty, 2005; Swobok, 2004 ; Lyndsay et van Eeden, 2003; James et Wood, 2003; Amland et Vaga, 2002). Selon Kaner (2004) n'importe quelle activité de test nécessite la créativité et les compétences pour être efficace, incluant le TS. Selon ce même auteur, les cas de tests scénarisés peuvent limiter la créativité de testeur et la transformer en un robot exécutant les tâches demandées sans aucune réflexion critique, surtout, si le rendement du testeur est évalué par le nombre de cas de test exécutés et par les erreurs détectées. Dans une telle situation, le testeur se concentre sur l'exécution des procédures de test et évite la déviation et l'improvisation.

Dans le but de montrer les effets négatifs de TS sur la créativité du testeur, Kaner et son équipe (Kaner et Bach, 2005) ont élaboré plusieurs expériences et recherches dans les laboratoires de Florida Institut sur des souris, dont les démonstrations sont disponibles dans (Kaner et Bach, 2005). Ces recherches ont prouvé que les procédures de test pourraient empêcher le testeur de voir d'autres problèmes non spécifiés dans les procédures, mais qui peuvent apparaître pendant l'exécution, à cause du fait que le testeur se concentre seulement sur les résultats identifiés dans les procédures de tests. Cela est connu en psychologie sous le terme anglophone «Inattentional Blindness» (Mack et Rock, 2000). Selon Kaner (2004) le TE aide le testeur à apprendre de nouvelles méthodes et stratégies de test en lui permettant d'améliorer sa capacité d'analyse et de réflexion critique. Selon les constations de Petty (2005) la possibilité d'apprentissage dans le TE est plus grande que celle en TS.

Nous concluons que le TE est approprié dans les projets de test ayant des testeurs compétents et experts. Tandis que le TS est plus approprié dans les projets de test que ont un nombre limité de testeurs experts et plusieurs non expérimentés.

7.4.2 La culture de l'organisation

La culture de l'organisation pourrait fortement influencer le choix de l'approche de test. Selon Craig et Jaskiel (2002) un processus de TS ne pourrait pas être mis en place s'il n'existait pas un processus de développement formel et bien structuré qui supporte le projet de test. Souvent, les entreprises qui utilisent les bonnes pratiques et les méthodes disciplinées de développement favorisent plus l'utilisation de TS qui convient plus avec leur méthodologie de développement. Tandis que les entreprises qui possèdent des processus immatures ou chaotiques de développement ne pourraient pas utiliser le TS. Ces entreprises favorisent souvent des méthodes de test informelles comme le TE (Lyndsay, van Eeden, 2003, Itkonen et Rautiainen, 2005).

Nous concluons que le TS est plus approprié dans les projets de test des entreprises favorisant la discipline. Tandis que le TE est plus approprié dans les entreprises qui ont des processus de développement immatures et qui s'appuient sur les compétences et la créativité du personnel pour mener les activités de développement et éviter le chaos.

7.5 Comparaison selon la productivité

Dans cette section nous allons comparer les deux approches de test, le TS et le TE selon le facteur de «la productivité ». Nous discutons la productivité en termes du nombre de défauts détectés et de l'importance de défauts détectés.

Depuis son apparition dans le domaine du test, le TE s'est fait promouvoir comme une méthode efficace. Selon Bach (2003) le TE pourrait être plus productif que le TS. Cependant, il n'y a aucune preuve jusqu'à maintenant dans la littérature prouvant cette productivité, à part quelques anecdotes et expériences vécues des praticiens.

Théoriquement, l'efficacité pourrait être perçue autrement que basée seulement sur la base du compte des défauts trouvés. Selon Copeland (2004) le TS est plus avantageux que le TE, du fait qu'il pourrait s'introduire tôt dans le cycle du développement du logiciel. En effet, dans les vues préventives actuelles, le TS peut commencer dès le début de projet du développement. Par la suite, il pourrait détecter les erreurs dans la conception et les exigences avant que ces dernières ne se transforment en des défauts dans le logiciel. Par contre, le TE

ne pourrait s'introduire qu'après le codage du logiciel. De ce point de vue, nous pouvons conclure que le TS est plus efficace que le TE, vu sa capacité de prévenir les défauts. D'un autre point de vue, selon (Copland, 2004; Kaner, 2003) les cas de tests scénarisés deviennent «fatigués» c'est-à-dire incapables de détecter de nouveaux défauts dans les cycles ultérieurs de test. Par conséquent, le TE pourrait être plus efficace dans cette situation.

Les auteurs Itkonen, Rautiainen (2005) ont tenté de collecter des données quantitatives pouvant leur donner une idée de la productivité de TE. Malheureusement, les données collectées n'étaient pas fiables. S'ajoute à ce fait, l'absence des résultats de TS pouvant servir comme une référence de comparaison. Dans les autres études de cas que nous avons proposées dans la revue de littérature, les praticiens ont tous rapporté que leur expérience dans l'utilisation de TE comme une approche primaire de test a été fructueuse et réussie. Cependant, ils n'ont pas présenté des données quantitatives sur l'efficacité réalisée et les résultats obtenus.

Quant à notre expérience qui s'est déroulée dans les laboratoires informatiques de l'UQÀM, nous n'avons pas pu tirer des conclusions fiables et généralisées, à cause des limites du contexte de l'expérience. Cette limite est causée par la taille du programme utilisé dans l'expérience, choisi pour faciliter la tâche des sujets et éviter les résultats nuls. Le contexte ne nous a pas permis d'aborder l'activité de test d'une façon professionnelle. Le manque d'expérience des sujets dans le domaine de test du logiciel s'ajoute aussi aux limitations de contexte. Ces limitations ont influencé les résultats de l'expérience. Cela est apparu clairement dans les résultats de TS obtenus. Quoique nous ayons proposé aux sujets les mêmes scénarios de cas de test, nous avons constaté que les sujets ont interprété les résultats observés différemment et ils ont eu de la difficulté à classer les défauts détectés correctement. Nous avons d'ailleurs dû les reclasser après l'expérience.

Les résultats recueillis de cette étude empirique nous a permis de conclure que le TE est plus productif en terme de nombre des défauts détectés que le TS. Ainsi, nous avons conclu que le TE pourraient détecter des défauts plus importants, point de vue gravité, que le test scénarisé.

7.6 Discussion et synthèse

Dans ce chapitre, nous avons comparé les deux approches de test, le TE et le TS, selon les facteurs du cadre de comparaison que nous avons élaboré. Dans cette section, nous allons récapituler les résultats et conclure. Le tableau ci-bas récapitule les résultats de notre analyse comparative. Il inclut les facteurs principaux qui doivent être pris en compte pendant la sélection de l'approche de test pour éviter de proposer une solution inadéquate.

En fait, ce tableau constitue un guide de sélection de l'approche de test, parmi les deux discutés dans ce travail de recherche à savoir : le TE et le TS. Alors, ce guide identifie comment chaque facteur de contexte de test s'applique à chacune des deux approches de test. En analysant chaque facteur d'un contexte de test particulier suivant ce guide (Tableau 7.1) nous pouvons savoir si le contexte est favorable pour utiliser le TE à la place de la méthode traditionnelle scénarisée. Ce guide tente de baliser une démarche d'analyse pouvant être intégrée à la réflexion stratégique des entreprises pendant la sélection de l'approche de test. Ainsi, il pourrait aider à comprendre les apports et les besoins de TE, ce qui va permettre d'assimiler et adopter l'approche.

Tableau 7.1 : Le guide de sélection de l'approche de test

Facteurs	Test scénarisé	Test exploratoire
1. Caractéristiques d'utilisation		
Les raisons d'utilisation	La stabilité de projet de développement, le besoin de documenter et mesurer l'activité de test	L'instabilité de projet de développement, l'absence des exigences
Les caractéristiques du logiciel		
La taille du logiciel	Les grands logiciels	Les logiciels petits et moyens
La complexité du logiciel	Plus approprié	Dépend des compétences existant dans le projet de test
La criticité du logiciel	Exigé	Ne devrait pas être utilisé
Le type d'environnement d'affaire	Stable, évolutif, sous contrat et n'importe quel environnement qui nécessite une documentation détaillée des tests.	Dynamique.
Les ressources financières	Ressources importantes	Ressources raisonnables ou limitées
Le temps de test disponible	Temps considérable requis	Peu de temps requis
2. Caractéristiques de gestion		
La planification	Rigoureuse, classique et certaine	Adaptative, continue, non rigoureuse
Le contrôle et le suivi de progression de test	Mesurable	Estimé
La relation avec le client	Souhaitée	Essentielle, peut améliorer la qualité du test
La communication dans le projet de test	Souhaitée	Essentielle, la qualité de test effectué dépend de la qualité de communication existante dans le projet de test
3. Caractéristiques techniques		
Les activités de test	Répétitives, auditables	Créatives, adaptatives
L'oracle de test	Fixe, dérivé à partir des exigences du logiciel et les standards	Variable, dérivé à partir des prévisions du testeur
Les risques du logiciel	La couverture de tous les risques est assuréc	La couverture de tous les risques n'est pas assurée

Facteurs	Test scénarisé	Test exploratoire
3. Caractéristiques techniques		
La couverture de test	Mesurée	N'est pas assurée ni mesurée
4. Caractéristiques du personnel		
Les caractéristiques du testeur	Testeurs compétents et experts	Nombre limité d'experts et plusieurs non expérimentés
La culture de l'organisation	Disciplinée	Immature
5. Productivité		
Le nombre des défauts détectés	Moins productive que le TE	Plus productive que le TS
L'importance des défauts détectés	Moins importants que ceux qui pourraient être détectés avec le TE	Importants et critiques s'il est exécuté par des personnes compétentes

Or, les facteurs identifiés n'ont pas tous le même poids ni la même importance dans le contexte de test. Nous avons constaté d'après notre analyse comparative que la couverture de test est le facteur le plus important dans le contexte de test. Du fait que les autres facteurs sont inter-reliés d'une manière ou d'une autre avec la couverture du test. Aussi, nous avons constaté que les qualifications et les compétences existantes dans les projets de test pourraient influencer le choix de l'approche de test. Alors, dans les projets de test où les qualifications personnelles sont insuffisantes, il apparaît difficile d'utiliser le TE. Nous pouvons conclure que le TE pourrait remplacer le TS dans n'importe quel projet de test où le contrôle de la couverture ne constitue pas une exigence et où les compétences et les qualifications du personnel aident à utiliser le TE.

Pourtant, il est difficile de cerner un contexte où une seule approche de test parmi le TS et le TE conviendrait. À cet effet, nous croyons qu'une approche hybride peut convenir mieux, où certaines parties du logiciel pourraient être testées en utilisant le TS et d'autres en utilisant le TE. Ainsi, la méthodologie de test pourrait bénéficier des avantages de chacune des deux approches.

7.7 Conclusion

Au cours de ce chapitre, nous avons comparé et analysé les deux approches de test selon le cadre comparatif de comparaison que nous avons élaboré dans le chapitre précédent. Nous sommes revenues sur les résultats de l'étude empirique que nous avons menée dans le cadre de ce travail de recherche, afin d'évaluer empiriquement la productivité de test exploratoire, en termes du nombre et de l'importance de défauts détectés. L'analyse comparative selon les facteurs du cadre de comparaison nous a permis de ressortir les contextes favorables pour utiliser le TE comme une méthodologie primaire de test à la place de la méthode scénarisée habituelle. En terminant, nous avons élaboré un guide de sélection de l'approche de test. Ce guide récapitule les résultats de l'analyse comparative et tente de baliser une démarche d'analyse pouvant être intégrée à la réflexion stratégique des entreprises pendant la sélection de l'approche de test.

CHAPITRE VIII

ANALYSE DES RÉSULTATS

Ce chapitre présente une analyse des résultats de l'analyse comparative présentée au chapitre précédent et les résultats de l'étude empirique présentée dans le chapitre V. Ce chapitre fait aussi ressortir la contribution de l'étude et les pistes potentielles de recherche.

8.1 Analyse des résultats théoriques

L'analyse comparative du TE et du TS que nous avons effectuée dans le chapitre précédent nous a permis d'identifier les facteurs de contexte pouvant influencer le choix de l'approche de test. Nous avons identifié comment chaque facteur s'applique dans le cadre de chacune des deux approches. À cet effet, n'importe quel contexte de test pourrait être analysé selon les facteurs identifiés dans le guide de sélection de l'approche de test (tableau 7.1). Cela permet de savoir a priori l'approche la plus appropriée aux besoins du contexte de test.

Selon Copeland (2004) le TS pourrait être utilisé n'importe où l'objectivité, la répétitivité et l'auditabilité sont nécessaires (Section 2.2). Bach (2003) a mentionné que le TE pourrait être plus productif que le TS dans certaines situations. Or ces situations n'ont pas été identifiées dans aucune étude. Dans notre recherche, nous avons étudié ces situations, dans le cadre d'une analyse comparative théorique et empirique entre le TE et le TS selon un cadre de comparaison (figure 5.1) que nous avons développé afin de guider notre analyse comparative des deux approches. Par le biais de cette analyse comparative nous avons pu étudier, identifier et analyser les contextes qui favorisant l'adaptation et l'utilisation de TE comme une méthodologie indépendante de test. Notre étude a mis en évidence que d'autres facteurs que ceux identifiés par Copland (2004) pourraient favoriser l'utilisation de TS. Toutefois,

nous avons constaté que la couverture du test et les qualifications et l'expertise des personnels présents dans le contexte de test sont les deux facteurs les plus importants.

La première contribution de cette recherche est de présenter un guide de sélection de l'approche de test qui récapitule tous les facteurs du contexte de test et comment ils s'appliquent dans le cadre du TS et du TE. Ce guide pourrait être inclus dans la réflexion stratégique des entreprises pour sélectionner l'approche la mieux appropriée à n'importe quel contexte de test. Il constitue aussi un outil d'enseignement de TE qui peut aider les étudiants à mieux comprendre et à mieux différencier les contextes favorables pour utiliser chacune des deux approches.

8.2 Analyse des résultats empiriques

L'évaluation empirique de la productivité de TE n'a pas été bien abordée dans les travaux de recherches. Bach (2003) a proposé les résultats de ces expériences vécues comme preuves de la productivité du TE. Les auteurs Itkonen et Rautiainen (2005) ont collecté des données quantitatives de TE de deux petites entreprises qui utilisent le TE comme une méthode primaire de test. Toutefois, ces données ne sont pas fiables ni représentatives à cause de l'absence de données quantitatives de TS dans les deux cas, qui pourraient être considérées comme référence afin de prouver la productivité de TE. L'absence des études empiriques dans la littérature nous a obligé à considérer les résultats d'Itkonen et Rautiainen (2005) dans notre étude comparative.

L'originalité de notre étude empirique réside dans la conception innovatrice que nous avons choisie pour l'effectuer. Cette conception nous a permis d'évaluer plusieurs aspects:

- ❑ La productivité de TE en terme du nombre et de l'importance des défauts détectés pendant l'expérience, puis la comparaison des résultats de l'expérience avec les résultats rapportés par les auteurs Itkonen et Rautiainen (2005).
- ❑ L'effet de la technique de test (TE, TS) sur la performance des sujets pendant l'activité de test. De cette façon nous avons pu évaluer la capacité des sujets d'exécuter et de

reproduire les cas de tests de la même façon prévue par le concepteur (l'auteur de ce travail). Cela nous a permis d'explorer l'hypothèse de Kaner et Bach (2005) qui cite que le testeur dans le TS ne pourrait pas reproduire les cas de test de la même façon que leur concepteur. Nous avons aussi pu explorer la deuxième hypothèse de Kaner et Bach qui cite que le TS empêche le testeur d'apercevoir d'autres défauts que ceux qui sont documentés dans le cas de test. Ce phénomène est connu dans le domaine de psychologie sous le terme anglophone «Inattentional Blindness» (Mack et Rock, 2000)

L'analyse des résultats recueillis de l'expérience a montré que la moyenne des défauts détectés est de 9.5 dans une session de 45 minutes. Cette moyenne est proche de celle proposée par la recherche des auteurs Itkonen et Rautiainen (2005), soit 8.7 dans une session de 60 minutes.

En ce qui concerne l'importance des défauts détectés par le TE, nous avons conclu que plus que la moitié des défauts détectés ont été des défauts graves, tandis que les défauts fatals ont constitué 10% de total des défauts détectés. Les auteurs Itkonen et Rautiainen (2005) ont rapporté un pourcentage de 15% des défauts fatals détectés dans une session de 60 minutes. Ce pourcentage est proche de pourcentage que nous avons obtenu dans notre étude empirique, si nous prenons en considération la différence dans les programmes utilisés. Nous avons aussi étudié dans notre étude empirique l'influence de l'expertise de testeur sur les résultats de TE. À cet effet, nous avons étudié la relation entre le niveau de connaissance en Java et le nombre des défauts détectés. Notons que le niveau de connaissance en Java représente dans notre étude empirique l'expertise et les qualifications de l'expérimentateur. Alors, les résultats ont montré que la moyenne de défauts détectés croît en fonction de niveau de connaissance en Java.

En ce qui concerne la productivité du TE par rapport au TS nous avons conclu que le TE est plus productif que le TS, du fait que la performance des sujets dans le TE a été supérieure de celle réalisée dans le TS. Par la suite nous avons conclu que le TE a un effet positif sur le rendement des sujets en comparaison avec le TS. Ces résultats appuient les hypothèses de

Kancr et Bach (2005). Toutefois la limitation de contexte de l'étude empirique a affaibli la validité externe des résultats. De ce fait ces résultats ne pourront pas être généralisés.

Les résultats quantitatifs de cette étude empirique ainsi que sa conception constituent la deuxième contribution de notre travail de recherche. Nous pensons que les conclusions tirées de cette étude sont susceptibles de s'appliquer dans des répliques futures, et les chercheurs intéressés par l'évaluation de la productivité de TE empiriquement pourront réutiliser notre étude empirique comme ébauche pour leurs études.

8.3 Pistes potentielles de recherche

Le but de ce travail était d'évaluer empiriquement la productivité de TE et d'explorer les contextes qui favorisent son utilisation à la place de la méthode scénarisée habituelle. Suite à cette recherche, plusieurs avenues mériteraient d'être approfondies.

□ Enrichir le guide de sélection de l'approche de test

Nous avons considéré dans le développement du guide de sélection de l'approche de test les facteurs qui ont influencé les résultats de l'utilisation de TE dans l'industrie. Or avec la croissance de l'adoption et de l'adaptation du TE par les entreprises, d'autres facteurs pourront apparaître. L'essai de ce guide dans l'industrie pourrait engendrer aussi des feedbacks et des apports utiles. À cet effet, il serait intéressant de vérifier si d'autres facteurs peuvent s'appliquer et venir enrichir le guide.

□ Réplication de l'étude empirique dans un contexte industriel

Le contexte académique où s'est déroulée l'étude empirique a influencé négativement la validité externe de l'expérience et a constitué sa limite majeure. Pour dépasser cette limite, il serait intéressant de reprendre l'étude dans un contexte industriel en utilisant plusieurs projets de test. Ainsi l'évaluation de la productivité pourrait être faite sur deux étapes: pendant l'activité de test et après l'activité de test, c'est à dire dans le site de production de chacun des logiciels qui faisaient l'objet de cette étude.

□ **Explorer l'influence de l'expertise et les connaissances du domaine sur la qualité de TE**

Il serait intéressant d'étudier l'influence de l'expertise et les connaissances du domaine sur la qualité de TE effectué en termes du nombre des défauts détectés et de l'importance de ces défauts dans un contexte industriel en utilisant le nombre d'années d'expérience dans le domaine du test comme une métrique de l'expertise.

8.4 Conclusion

Au cours de ce chapitre, nous avons effectué une analyse et présenté une discussion des résultats de notre recherche, dans laquelle nous avons situé ses contributions au niveau théorique et empirique. Par la suite, nous avons présenté des pistes de recherche futures dans lesquelles nous avons identifié d'autres problématiques qui peuvent être utiles en vue d'initier des travaux futurs de recherches.

CONCLUSION

Nous avons étudié dans ce travail deux approches de test du logiciel: le test exploratoire (TE) et le test scénarisé (TS). La partie théorique de ce travail a eu comme but l'exploration des contextes du test où le TE pourrait remplacer le TS et être utilisé comme une méthodologie primaire du test. La partie empirique a eu pour objectif l'évaluation de la productivité de TE annoncée dans la littérature.

Après la définition du sujet de recherche, nous avons présenté une vue d'ensemble de TS et de TE successivement. Par la suite, nous avons présenté une revue de littérature de quelques études de cas et expériences d'utilisation de TE dans l'industrie du test. C'est ainsi que nous avons identifié les facteurs influençant l'adoption et l'adaptation de TE dans la pratique du test. Par la suite, nous avons présenté les résultats de l'étude empirique que nous avons menée dans les laboratoires de l'UQÀM. Puis, nous avons élaboré un cadre conceptuel de comparaison dans lequel nous avons identifié les dimensions principales de notre analyse comparative de TE et TS. Ce cadre pourra servir dans d'autres études comparatives des approches du test.

L'analyse comparative réalisée a permis de ressortir les facteurs contextuels favorables pour utiliser chacune des deux approches du test. Entre autres, nous avons montré que le TE n'est pas approprié dans les projets de test nécessitant que la couverture de test soit mesurée. Aussi, nous avons montré que la dépendance de TE à l'expertise et les qualifications du testeur rend difficile son utilisation dans les contextes où les qualifications et les compétences du personnel sont insuffisantes. À cet effet, Nous avons conclu que le TE pourrait remplacer le TS dans n'importe quel projet de test où le contrôle de la couverture ne constitue pas une exigence et/ou les compétences et les qualifications du personnel permettent d'utiliser le TE

Notre étude empirique a montré la productivité de TE en termes de nombre et l'importance des défauts détectés. Toutefois nous ne pouvons pas généraliser les résultats obtenus dans

cette étude à cause de la limitation du contexte ou s'est déroulé notre expérience. À cet effet nous reportons cette question à des travaux futurs de recherches, de préférence dans des contextes industriels.

Au niveau pratique, ce travail de recherche s'inscrit dans un courant de préoccupation des entreprises qui ont à la recherche des méthodes du test plus efficaces, qui pourraient s'adapter avec la culture rapide actuelle de développement du logiciel. Il est à noter que cette étude constitue une ouverture sur le développement d'un guide visant l'adaptation de TE dans l'industrie du test. Nous espérons que le guide de sélection de l'approche de test aide les entreprises et les praticiens à mieux choisir leur approche de test selon leur contexte du test.

Nos objectifs personnels étaient d'explorer les apports de TE et d'évaluer sa productivité fortement mise de l'avant par les praticiens de CDS. L'analyse comparative de TE et le TS nous a poussée à approfondir nos connaissances dans le domaine du test logiciel, pour que nous puissions identifier les apports et les lacunes de chacune des deux approches. Ce travail nous a permis de découvrir l'importance de l'activité du test dans le processus de développement logiciel. Cela nous a montré les différents côtés de test du logiciel, ses défis, son côté quasi artistique, mathématique et critique. Bref, nous avons trouvé un autre domaine d'intérêt, outre que la programmation et le développement

ANNEXE A

CADRE DE BASILI ADAPTÉ À LA RECHERCHE EXPLORATOIRE

Les tableaux présentés dans cette annexe récapitulent les phases du projet de recherche selon le cadre de Basili à la recherche exploratoire adapté par Abran et al (1999).

1. Définition			
Motivation	Portée	Objectif	Utilisateurs
1. Explorer les apports de TE 2. Explorer l'ampleur de la divergence entre les deux vues 3.Évaluer la productivité de TE 4.Élaborer un guide de sélection de l'approche de test	Comparaison théorique et empirique de TE et TS selon un procédé de test boîte noire	Répondre à la question principale de recherche: Est-ce que le TE pourrait remplacer le test scénarisé ? Si oui dans quel contexte ?	- Les chercheurs et les praticiens intéressés a l'étude du TE - Les entreprises désirant mettre en oeuvre ces approches de test

2. Planification du projet		
Les étapes	Les intrants	Les livrables
1. Proposer d'un modèle du processus de TS 2. Élaborer un cadre de comparaison 3. Faire l'étude comparative empirique de TE et TS 4. Faire l'étude comparative de TE et TS en se basant sur le cadre élaboré et les résultats empirique 5. Analyser et discuter des résultats	La norme IEEE 829	- Un cadre conceptuel de comparaison des approches de test - Les résultats quantitatifs de l'étude empirique - Un guide d'évaluation des facteurs de contexte de projet de test pour le choix d'une approche de test

3. Opération		
Analyse des documents	Feedback des praticiens	Modèles proposés
1. Identifier les facteurs qui ont influencé l'adoption et l'adaptation de TE dans l'industrie 2. Analyser l'étude comparative de Boehm et Turner 3. Étudier et analyser les connaissances théoriques de test du logiciel 4. Étudier et analyser les apports et les mécanismes de TE.		Cadre conceptuel de comparaison des approches de test qui inclut les cinq dimensions à savoir: 1. Les caractéristiques d'utilisations; 2. Les caractéristiques du logiciel à tester; 3. Les caractéristiques de gestion; 4. Les caractéristiques du personnel; 5. La productivité. Proposition d'un guide de sélection de l'approche de test

4. Interprétation		
Contexte d'interprétation	Extrapolation des résultats	Travaux futurs
<p>1. La comparaison théorique et empirique des deux approches de test : le TE et le TS a été réalisée</p> <p>2. L'analyse comparative de TE et TS selon les facteurs de notre cadre conceptuel de comparaison a permis d'identifier les contextes favorables pour utiliser le TE à la place de TS</p> <p>3. L'analyse comparative a permis d'élaborer un guide de sélection de l'approche de test</p> <p>4. Cette recherche pourrait contribuer à mieux comprendre le TE. Elle facilite l'adoption de TE au sein des entreprises qui oeuvrent dans le domaine du développement</p>	<p>- Les résultats de l'étude empirique sont limités. Ils dépendent du contexte académique où s'est déroulée l'expérience.</p> <p>- L'analyse comparative entre le TE et le TS est associée au cadre de comparaison élaboré dans ce travail de recherche et les résultats de l'étude empirique. À cet effet, cette analyse est limitée et en partie subjective. Elle dépend des connaissances et de l'expérience de l'auteure dans le domaine de test du logiciel.</p>	<p>- Réplication de l'étude empirique et comparative de TE et TS dans un environnement industriel</p> <p>- L'étude de l'influence des connaissances du domaine et l'expertise de testeur sur les résultats de TE</p> <p>- L'amélioration du guide de sélection de l'approche de test élaboré dans ce travail de recherche</p>

ANNEXE B

PLAN DE TEST IEEE829 (ADAPTÉ ET TRADUIT DE IEEE 829,1998)

1. Identificateur de plan de test

- Identificateur unique de document qui pourrait le distinguer des autres documents.

2. Introduction

- l'introduction pourrait inclure les paragraphes suivants :
 - Description du logiciel sous test : ce paragraphe donne un aperçu du logiciel, ses opérations, maintenance, histoire, son code et toute autre information pertinente;
 - Une liste de références à d'autres documents utiles pour la compréhension du plan de test. Selon la norme IEEE 829, les références aux documents suivants, s'ils existent, doivent être mentionnées:
 - ☐ Autorisation du projet ;
 - ☐ Plan du projet de développement ;
 - ☐ Plan d'assurance qualité ;
 - ☐ Plan de gestion de configuration ;
 - ☐ Politique pertinente de la compagnie et du client ;
 - ☐ Normes pertinentes de la compagnie, du client ou de l'industrie ;

3. Items de test

- Identifie les items à tester incluant leur version/niveau de révision

4. Caractéristiques à tester

- Identifie les caractéristiques à tester, telles que fonctionnalité, performance, sécurité, portabilité, etc.

5. Caractéristiques qui ne doivent pas être testées

- Identifie les caractéristiques qui ne vont pas être testées, ainsi les raisons de ce fait

6. Approche

- Propose la stratégie globale de test afin d'assurer que tous les items et leurs caractéristiques seraient testés adéquatement

7. Critère de passage/échec

- Définit le critère de passage et d'échec de test de chaque item.

8. Critère de suspension et conditions de reprise

- Définit les circonstances dans lesquelles le test pourrait être arrêté et les conditions pour qu'il reprenne (défauts critiques qui nécessitent la re-conception, environnement de test incomplet, etc.)

9. Livrable du test

- Identifie les documents de test, ainsi que les autres livrables devant être produits au cours du projet de test (ex. spécifications de conception de test, spécifications de cas de test, spécifications de procédure de test, registre de test, rapport d'incident de test, rapport de synthèse de test, matrice de traçabilité, etc.)

10. Tâche de test Identifie les tâches de test et l'interdépendance entre elles ainsi que la durée et les ressources requises pour les accomplir.

11. Besoins environnementaux

- Spécifie l'environnement requis pour accomplir l'activité de test incluant, matériel, logiciel, outils, etc.

12. Responsabilités

Identifie la responsabilité et la tâche de chaque membre de l'équipe de test

13. Besoins en personnel et en formation

- Les personnes et les qualifications nécessaires pour réaliser le plan

14. Calendrier

- Spécifie les étapes importantes dans le plan de projet de développement ainsi que les items qui devraient être transmis à chaque étape

15. Risques et contingences

- Identifie les risques qui peuvent empêcher le suivi du plan et les mesures à prendre pour les surmonter

16. Approbation

ANNEXE C

SOIRÉE DE TESTS

Document remis aux participant(e)s

L'objectif premier de cet exercice est d'augmenter votre niveau d'expérience dans l'exécution de tests préparés par d'autres, et dans le domaine des tests exploratoires. Pour ce faire, vous utiliserez d'abord, jusqu'à 20:00, l'approche des tests exploratoires, à l'aide des directives données dans la section suivante. Dans la deuxième partie de la soirée, vous exécuterez des tests scénarisés qui vous seront remis à 20:00.

Dans les deux cas, vous prendrez en note, sur les formulaires ci-joints (voir Annexe D), les défauts que vous constaterez. Par la suite, vous complétez les rapports demandés en répondant aux questions proposées. Toutes ces informations serviront de base à un travail de maîtrise sur les différences entre le TE et le test scénarisé

Quelques directives et informations pour exécuter le TE

Définition du programme

Il s'agit d'un gestionnaire simple de messages. Chaque message contient les informations suivantes: émetteur, destinataire, sujet du message, texte du message et une information supplémentaire qui indique si le message a été lu ou non, éliminé ou non. Pour chaque message le logiciel attribue un numéro automatiquement, supérieur à 100. Le programme utilise un tableau de taille 10 pour gérer les messages. Le programme doit afficher un message un message d'erreur si on tente de générer plus de 10 messages.

Les directives à utiliser

Nous proposons certaines techniques qui peuvent vous aider dans vos tests exploratoires.

Vous pouvez choisir une ou plusieurs de ces techniques. Vous n'êtes pas obligés de les appliquer strictement, et vous avez toujours la possibilité d'introduire vos propres techniques. Cependant, rappelez-vous que le but de l'exercice est de découvrir le plus possible de bogues importants, de façon à améliorer la qualité du logiciel.

- ❑ Exécution du programme en utilisant des données valides (parmi les choix affichés dans le menu principal) pour avoir une idée sur ses fonctions et ses caractéristiques principales
- ❑ Suivez votre intuition et explorez le programme, si vous avez une idée sur les types d'erreurs qu'il peut inclure.
- ❑ Essayez de générer des questions sur la capacité du programme d'accomplir certaines fonctions que vous sont apparues essentielles, mais toujours dans le cadre de la description ci-dessus. Essayez ensuite de transformer chaque question en un jeu d'essai (cas de test).
- ❑ Générez différents scénarios d'utilisation de logiciel. Ensuite essayez d'explorer les aspects de vos scénarios, par exemple, utilisez différentes valeurs d'entrée, surtout les valeurs extrêmes (limites) pour le même scénario.
- ❑ Vérifiez les messages d'erreurs du programme, autrement dit vérifiez s'ils se déclenchent au bon moment et sous les bonnes conditions.
- ❑ Choisissez une variable, puis essayez de tracer son flux dans le programme. Les méthodes qu'elles l'utilisent ainsi que ses interactions avec d'autres variables. Ensuite utilisez ces informations pour interférer avec la variable.
- ❑ Choisissez une tâche parmi les fonctionnalités du programme et essayez de décrire étape par étape comment elle est accomplie et manipulée dans le logiciel puis essayez d'improviser et de concevoir des techniques pour la tester (par exemple en utilisant des valeurs d'entrées qui peuvent pousser la fonction dans d'autres chemins).
- ❑ Utilisez un diagramme d'états pour décrire les différentes actions et transitions que

l'application peut prendre pour toutes les entrées possibles. Essayez ensuite de trouver les contradictions dans le modèle.

La procédure

- Le travail doit être fait individuellement et aucun contact avec un(e) autre étudiant(e) n'est permis durant toute l'activité de test.
- Notez l'heure de début et de fin de vos tests exploratoire.
- Durant votre activité de test à chaque fois que vous trouvez un bogue, inscrivez les informations demandées dans la liste que vous a été remise. Vous devez décrire en détail l'erreur. Vous devez décrire brièvement comment vous avez réalisé qu'il s'agit d'une erreur par exemple l'absence d'un menu ou changement dans la valeur de sortie prévue, etc. Vous devez aussi classer l'erreur selon sa gravité. Ces informations sont aussi données avec le formulaire d'inscription des défauts.

ANNEXE D

RAPPORT DE LA SESSION DE TEST

Nom de l'étudiant(e):

- Comment évaluez-vous vos connaissances en Java ?

Niveau	Jamais vu	Introduction	Intermédiaire	Avancé	Expérimenté
Estimation					

- **Classification**

On classifie la sévérité des erreurs en trois niveaux :

- ☐ Fatale (F): l'application est inopérable complètement (Crash).
- ☐ Grave (G): l'application fonctionne, mais certaines fonctions sont inopérables ou certains résultats sont erronés
- ☐ Mineure (M): l'impact est mineur sur l'utilisation du système: ex: certains formats sont erronés, bien que les résultats soient corrects, ou encore les délais sont supérieurs à ce qu'on attend d'une telle application.

N'oubliez pas de prendre des notes sur les techniques d'exploration que vous utilisez.

Description de l'erreur	Classification

RÉFÉRENCES

Abran, Alain, Lucie Laframboise et Pierre Bourque. 1999. « *A Risk Assessment Method and Grid for Software Engineering Measurement Programs* ». Montréal : Université du Québec à Montréal.

Amland, Ståle, et Jarle Våga. 2002. « Managing High Speed Web Testing ». In *Software Quality and Software Testing in Internet Times*, sous la dir. de Meycrhoff, Dirk, Begona Laibarra, Rob Van Der Pouw Kraan et Alan Wallet, P. 23-30. Berlin : Springer.

Amland, Ståle. 2002a. «Exploratory Testing, Planning, Execution and Documentation». Version (1.20), www.testingeducation.org.

Amland, Ståle. 2002b. «Exploratory Testing Styles». Version (1.20). www.testingeducation.org.

Bach, James. 2007. « Rapid Software Testing ». Version (1.3.2), www.satisfice.com.

Bach, James, et Jonathan Bach. 2006. « Exploratory Testing Dynamics ». Version 1.6.

Bach, James. 2003. «Exploratory Testing Explained ».Version (1.3).

Bach, James, Bret Pettichord et Cem Kaner. 2002. *Lessons Learned in Software Testing*. New York: John Wiley & Sons.

Bach, James. 2001. «Exploratory Testing and Planning Myth». (Stickyminds.Com), 19 mars.

Bach, James. 1999. «General Functionality and Stability Test Procedure». www.satisfice.com

Bach, James. 1996. «Heuristic Test Strategy Model». www.satisfice.com

Bach, Jonathan. 2000. « Session Based Test Management ». *Software Testing and Quality Engineering*, novembre.

Basili, Victor, Richard Selby et David Hutchens. 1986. «Experimentation in Software Engineering». *IEEE Transactions on software engineering*, vol. 12, no. 7, p. 733-743.

Beedle, Mike, et al. 2001. «Manifesto for Agile Software Development». Consulté, janvier 2007, <http://agilemanifesto.org/>

Black, Rex. 1999. *Managing the Testing Process*. Redmond (Washington): Microsoft Press.

Beizer, Boris. 1995. *Black Box Testing: Techniques for Functional Testing of Software and Systems*. New York: John Wiley & Sons

Beizer, Boris. 1990. *Software Testing Techniques*, 2^e éd. New York: Van Nostrand Reinhold.

Boehm, Barry W, et Richard Turner. 2004. *Balancing Agility and Discipline*. Boston: Addison-Wesley

Boehm, Barry W. 1981. *Software Engineering Economics*. Upper Saddle River (N.J): Prentice-Hall

Bolton, Michael, James Bach et Cem Kaner. 2005. « Rapid Testing Explained ». International Quality Conference 2005. Toronto, octobre.

Copeland, Lee. 2004. *A Practitioner's Guide to Software Test Design*. Boston: Artech House Publishers.

Craig, Rick, et Stefan Jaskiel 2002. *Systematic Software Testing*. Boston: Artech House Publishers.

Hendrickson, Elizabeth. 2005. « Agile Testing ». Video de Google, www.google.com.

Hetzel, William C. 1988. *The Complete Guide to Software Testing*, 2^e éd. New York: John Wiley & Sons

Hoffman, Douglas. 1999. «Heuristic Test Oracles». *Software Testing and Quality Engineering*, mars/avril, www.softwarequalitymethods.com/Papers/STQE%20Heuristic.pdf

Itkonen, Juha, et Kristian Rautiainen. 2005. « Exploratory Testing: A Multiple Case Study ». *Empirical Software Engineering: 2005 International Symposium*, novembre.

IEEE829. 1998. *Standard for Software Test Documentation*. New York: IEEE press

IEEE729. 1983. «IEEE Computer Society: IEEE Standard Glossary of Software Engineering Terminology».

IEEE610. 1990. «IEEE Standard Glossary of Software Engineering Terminology».IEEE STD610.2.

James, David, et Bill Wood. 2003. «Applying Session Based Testing to Medical Software». *Medical Device& Dignostic Industry*, mai.

Jones, T.Capers. 1998. *Estimating Software Costs*. New York: McGraw-Hill, p.554.

Kan, Parrish et Manlove. 2001. «In-process Metrics for Software Testing» *IBM Systems Journal*, vol. 40, no 01

Kaner, Cem. 2006. «Exploratory Testing after 23 Years» Conference of the Association for Software Testing, Indianapolis (U.S): www.Testingeducation.com

Kaner, Cem, et James Bach. 2005. « Scripting: An Industry Worst Practice ». *Black Box Testing Course*, www.Testingeducation.com

Kaner, Cem. 2004. «The Ongoing Revolution in Software Testing». Software Test & Performance Conference (Baltimore, MD, 7-9 décembre 2004). www.Testingeducation.com

Kaner, Cem, et James Bach. 2004. «The Nature of Exploratory Testing». Software Testing Day: University Tampere, Finland, consulté le 15/01/2007

Kaner Cem .2003. «What is a Good Test Case? ». STarEast Conference, May 2003 <http://www.kaner.com/pdfs/GoodTest.pdf>

Kaner, Cem, et Andy Tinkham. 2003a. «Exploring Exploratory Testing». STarEast Conference on Software Testing Analysis and Review (Orlando, FL, May 12-16, 2003)

Kaner, Cem, et Andy Tinkham. 2003b. «Learning Style and Exploratory Testing». Pacific Northwest Software Quality Conference (Portland, OR, October 13-15, 2003).

Kaner, Cem, Jack Falk et Hung Quoc Nguyen. 1999. *Testing Computer Software*. 2^e éd. New York: John Wiley and Sons.

Kaner, Cem. 1997. « *The Impossibility of Complete testing* ». *Low of Software Quality Column. Software QA*, vol. 04, no. 04, <http://www.kaner.com/pdfs/imposs.pdf>

Kaner, Cem.1996. «Software Negligence and Testing Coverage» *Software QA quarterly*, vol. 02, no 03, p. 18, <http://www.kanecr.com/coverage.htm>

Kanecr, Cem. 1988. *Testing Computer Software*. 1^{er}édi. New York: McGraw-Hill.

Kohl, Jonathan. 2005. «Exploratory Testing on Agile Teams». *Informit.Com*, 18 Novembre <http://www.informit.com/articles>

Lindsay, James, et Neil Van Eeden. 2003. «Adventures in Session Based Testing ». Version 1.2, <http://www.workroom-productions.com/papers/AiSBTv1.2.pdf>

Lott, Christopher, et Rombach Dieter. 1997. « Repeatable software engineering experiments for comparing defect detection techniques » *Empirical Software Engineering*. vol. 01, no 03, p. 241-277.

Mack, Arien, et Irvin Rock. 2000. «Inattentional Blindness». Cambridge (MA): Bradford Books/MIT press.

Marick, Brian. 2003. « Agile Methods and Agile Testing ».*Software Testing and Quality Engineering*, vol. 03, no 05.

Myers, Glenford J. 1979. *The Art of Software Testing*, 1^{ère} édi. New York: John Wiley

Osterweil, Leon. 1996. «Strategic directions in software quality » *ACM Computing Surveys*, vol. 04, no 04.

Perry, William E. 2000. *Effective Methods for Software Testing*, 2^e édi. John Wiley and Sons

Petty, Kenn. 2005. «Reflections on the Use of Session Based Exploratory Testing as the Primary Test Methodology for Software in an Agile Environment» *Indianapolis Workshops on Software Testing*, http://www.iwst2007.com/images/Reflections_on_the_use_of_Session-Based_Exploratory_Testing_in_an_Agile_Environment.doc

Pettichord, Bret. 2002. « Agile testing: What is it? Can it work? ». Consulté, Janvier, 2007 www.Pettichord.com.

Pressman, Roger. 1997. *Software Engineering: A Practitioner's Approach*, 4^e édi. New York: McGraw-Hill.

Pyhäjärvi, Maaret, Kristian Rautiainen et Juha Itkonen. 2003. «Increasing understanding of the modern testing perspective in software product development projects » *IEEE Computer Society*. Proceedings of the Hawaii International Conference on System Sciences

Royce, Wiston. 1970. «Managing the Development of Large Software Systems: Concepts and Techniques» Reprinted in 9th International Conference in Software Engineering. Los Alamitos (CA), IEEE Computer Society Press. p. 328-338.

SWEBOK. 2004. « Guide to the Software Engineering Body of Knowledge » *IEEE Computer Society*. Version 2004, <http://www.swcbok.org/>

Thompson, Neil. 2003. «Best Practices and Context-Driven: Building a Bridge». International Conference on Software Testing: Analysis and Review. StarEast, Florida. StickyMinds Magazine.

Whittaker, James A. 2003. *How to Break software: A Practical Guide to Testing*. Boston: Addison Wisely.

Winer, Ben James. 1971. *Statistical Principles in Experimental Design*, 2^e édi. New York: McGraw Hill.

Wood, Murray, Marc Roper, Andrew Brooks et James Miller 1997. « Comparing and Combining Software Defect Detection Techniques: A Replicated Empirical Study ». ACM SIGSOFT Proceeding on the Foundations of Software Engineering N°5. Zurich, SUISSE (22/09/1997), vol. 22, no 6. New York: Springer-Verlag (ACM Press)